

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The letters are black on a white background. The overall shape is roughly rectangular, with a central vertical column of 'Y's and 'S's. The left and right sides are composed of 'S's, with some 'Y's interspersed. The top and bottom edges are also composed of 'S's. The central vertical column is the tallest, with 'Y's at the top and 'S's at the bottom. The overall effect is a complex, stylized letterform that resembles a 'W' or a complex 'S'.

SSSSSSSS	YY	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	FFFFFFFFFF	LL	
SSSSSSSS	YY	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	FFFFFFFFFF	LL	
SS	YY	YY	SS	DD	DD	FF	FF	LL	
SS	YY	YY	SS	DD	DD	FF	FF	LL	
SS	YY	YY	SS	DD	DD	FF	FF	LL	
SS	YY	YY	SS	DD	DD	FF	FF	LL	
SSSSSSS	YY	YY	SSSSSSS	DD	DD	FFFFFFFF	FFFFFFFF	LL	
SSSSSSS	YY	YY	SSSSSSS	DD	DD	FFFFFFFF	FFFFFFFF	LL	
SS	YY	YY	SS	DD	DD	EE	FF	LL	
SS	YY	YY	SS	DD	DD	EE	FF	LL	
SS	YY	YY	SS	DD	DD	EE	FF	LL	
SS	YY	YY	SS	DD	DD	EE	FF	LL	
SSSSSSSS	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FF	FF	LLLLLLLLLL	LL	
SSSSSSSS	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FF	FF	LLLLLLLLLL	LL	

```

SSSSSSSS DDDDDDDD LL
SSSSSSSS DDDDDDDD LL
SS        DD        DD LL
SS        DD        DD LL
SS        DD        DD LL
SS        DD        DD LL
        SSSSSS DD        LL
        SSSSSS DD        LL
                SS DD        LL
                SS DD        LL
                SS DD        LL
                SS DD        LL
SSSSSSSS DDDDDDDD LLLLLLLLLL
SSSSSSSS DDDDDDDD LLLLLLLLLL

```


{ Version: 'V04-001'

{*****
{* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
{* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
{* ALL RIGHTS RESERVED.
{*

{* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
{* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
{* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
{* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
{* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
{* TRANSFERRED.
{*

{* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
{* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
{* CORPORATION.
{*

{* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
{* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
{*
{*
{* *****

{++

{ FACILITY: VAX/VMS System Macro Libraries

{ ABSTRACT:

{ This file contains the SDL source for all operating system control
{ blocks, from F to L. That is, all control blocks from FAA to LZZ.

{ ENVIRONMENT:

{ n/a
{--

{ AUTHOR: The VMS Group

{ CREATION DATE: 1-Aug-1976

{ MODIFIED BY:

{ V04-001 SRB0145 Steve Beckhardt 6-Sep-1984
{ Moved CDRP\$L_VAL9 into regular CDRP out of long CDRP.

{ V03-099 MSH0063 Michael S. Harvey 31-Jul-1984
{ Eliminate GSNAMOFF cell from KFE.

{ V03-098 ACG0440 Andrew C. Goldstein, 23-Jul-1984 11:49
{ Add ref count and classification valid flag to ORB in the FCB;
{ add FCB\$L_CACHELKID for file cache interlocks

V03-097 MSH0063 Michael S. Harvey 20-Jul-1984
Add symbol for maximum length of KFE.

V03-096 CDS0006 Christian D. Saether 15-July-1984
Add another pool to F11BC. Add FCB\$\$_DIRINDX.

V03-095 RAS0319 Ron Schaefer 29-Jun-1984
Add new structure to logical names structures:
LNMC -- the logical name table name cache structure.

V03-094 ACG0432 Andrew C. Goldstein, 9-Jul-1984 21:35
Add JIB\$\$_ORG_BYTLM and JIB\$\$_ORG_PBYTLM fields

V03-093 MSH0061 Michael S. Harvey 5-Jul-1984
Add EXEONLY image flag to the KFE.

V03-092 MSH0058 Michael S. Harvey 27-Jun-1984
Define a new cell in the KFE to hold the length of a
string which will be used to more accurately specify
which memory a set of global sections for the KFE was
created in.

V03-091 MSH0057 Michael S. Harvey 25-Jun-1984
Define new KFE bit used to propagate some context across
the INSTALL REPLACE command.

V03-090 CDS0005 Christian D. Saether 9-May-1984
Correct longword misalignment introduced by CDS0004.

V03-089 MSH0041 Michael S. Harvey 2-May-1984
Add image alias limit constants to \$IHDEF.

V03-088 CDS0004 Christian D. Saether 19-Apr-1984
Changes to FCB.

V03-087 MSH0032 Michael S. Harvey 12-Apr-1984
Add GSD\$\$_ORB for all types of GSD. This cell will
be used to locate the Object Rights Block associated
with a global section.

V03-086 ADE0001 Alan D. Eldridge 11-Apr-1984
Add IND\$\$_INISHR and IHA\$\$_INISHR.

V03-085 TMK0005 Todd M. Katz 11-Apr-1984
Add KFES\$\$_ORB to \$KFEDEF. This cell will contain the address
of the Object Rights Block associated with a Known File Entry.

V03-084 ACG0414 Andrew C. Goldstein, 9-Apr-1984 11:17
Add IOCSV\$_ALLOC bit

V03-083 KPL0006 Peter Lieberwirth 7-Apr-1984
Add IOCSV NO TRANS to \$IOCDEF. This flag tells
IOCSTRANDEVNAM that logical name translation is
unnecessary because the caller already did the
translation.

V03-082 TMK0004 Todd M. Katz 07-Apr-1984
Change LNMTH\$L CHP to LNMTH\$L ORB within \$LNMSTRDEF to reflect
the replacement of shareable logical name tables' CHIP
protection templates with Object Rights Blocks.

V03-081 EMD0074 Ellen M. Dusseault 06-Apr-1984
Define status bit, IRP\$V_KEY in IRP\$W_STS and
IRP\$V_KEY\$ESC which will contain an address of a
descriptor describing an encryption key.

V03-080 LMP0221 L. Mark Pilant, 26-Mar-1984 9:25
Move FCB fields around to define an ORB within the FCB.

V03-079 KPL0005 Peter Lieberwirth 23-Mar-1984
Fix \$IO8NNDEF to track Nautilus changes.

V03-078 SSA0021 Stan Amway 23-Mar-1984
Backed out SSA0017. Bit 15 of IRP\$W_STS conflicts with
a private definition and usage in DODRIVER. Added
comment to warn others of impending doom.

V03-077 WMC0077 Wayne Cardoza 23-Mar-1984
Add hash byte to GSD.

V03-076 ACG0408 Andrew C. Goldstein, 22-Mar-1984 23:04
Rearrange XQP base address cells so they form descriptors

V03-075 JWT0169 Jim Teague 22-Mar-1984
Bump image ident minorid. Long shareable image names
require larger entries in the shareable image list.

V03-074 MMD0267 Meg Dumont, 22-Mar-1984 17:05
Add ANSI 3 file length extension in HDR4 label.

V03-073 SRB0117 Steve Beckhardt 18-Mar-1984
Added LKB\$B_TSLT and LKB\$B_DLCKPRI to \$LKBDEF.
Added VAL9 and VAL10 to \$IRPDEF

V03-072 SSA0017 Stan Amway 9-Mar-1984
Add IRP\$V_QLEN as bit in IRP\$W_STS to indicate that
the device queue length in the UCB has been incremented to
account for the IRP. IOCIOPST conditionally decrements the
device queue length based on the setting of this bit.

V03-071 LMP0207 L. Mark Pilant, 9-Mar-1984 9:02
Add FCB\$V_BADACL to indicate that the ACL is present, but
should not be used in protection checks. This is to note
that the ACL has been corrupted.

V03-070 TMK0003 Todd M. Katz 09-Mar-1984
Add LNM\$W_HASH to \$LNMSTRDEF's definition of a logical
name translation block.

V03-069 ACG0399 Andrew C. Goldstein, 28-Feb-1984 16:13
Add \$IOCDEF - flag bits for IOC\$SEARCH

V03-068 CDS0003 Christian D. Saether 28-Feb-1984
Forgot some counters in F11BC.

V03-067 CDS0002 Christian D. Saether 28-Feb-1984
Add modules F11BDEF and F11BCDEF.

V03-066 RLRI08SS1 Robert L. Rappaport 27-Feb-1984
Add explicit PFN's to \$I08SSDEF for PCNTL, NIPACKETBUF, etc.

V03-065 JWO152 Jim Teague 27-Feb-1984
Make changes to \$ISDDEF and \$IHIDEF required for variable
length ISDs and long shareable image filenames.

V03-064 ROW0298 Ralph O. Weber 27-FEB-1984
Define IRP\$W_ENDMSG\$IZ, a field in the class driver CDRP
extension which holds the size of the most recent MSCP end
message.

V03-063 WHM0003 Bill Matthews 24-Feb-1984
Renamed IDB\$B_COMBO_VECTOR to IDB\$B_VECTOR now that
IDB\$B_COMBO_VECTOR_OFFSET has been added.

V03-062 RLRKDZ2 Robert L. Rappaport 22-Feb-1984
Fix \$KDZDEF and extend fill space following BIIIC register
definition so that the BIIIC registers occupy an entire
virtual page.

V03-061 KPL0004 Peter Lieberwirth 6-Feb-1984
Fix bug introduced in V03-058. Doubly-defined PERNEX.

V03-060 WHM0002 Bill Matthews 3-Feb-1984
Added field IDB\$B_COMBO_VECTOR_OFFSET. Additional support
for edit 51.

V03-059 LMP0188 L. Mark Pilant, 3-Feb-1984 16:22
Add support for a classification block in the FCB.

V03-058 KPL0003 Peter Lieberwirth 3-Feb-1984
Add \$I08NNDEF, IO space layout for Nautilus.

V03-057 MSH0004 Michael S. Harvey 2-Feb-1984
Redefine parts of global section descriptors to
accomodate longer global section names. The name
field will be variable length for local memory and
PFN-mapped global section descriptors, and will simply
be lengthened for the fixed length shared memory variety.

V03-056 HH0002 Hai Huang 31-Jan-1984
Redefine the mount list head in the JIB as a standard
two-word list head

V03-055 ROW0281 Ralph O. Weber 14-JAN-1984
Add IRP\$L_DUTUFLAGS and IRP\$W_DUTUCNTR to class driver CDRP
extension.

V03-054 ACG0387 Andrew C. Goldstein, 12-Jan-1984 14:28

Add job mount list head to JIB; get job type codes in proper order

V03-053 ACG0385 Andrew C. Goldstein, 11-Jan-1984 18:30
Make MAXDETACH and MAXJOBS JIB fields words

V03-053 LJK0257 Lawrence J. Kenah 5-Jan-1984
Reorder fields in JIB to speed up process creation time.

V03-052 ACG0385 Andrew C. Goldstein, 29-Dec-1983 13:34
Add JIB\$B_JOBTYPE field, remove JIB\$V_DISxxx flags

V03-051 WHM0001 Bill Matthews 27-Dec-1983
Added the field IDB\$B_COMBO_CSR_OFFSET. Replaced the field IDB\$B_VECTOR with IDB\$B_COMBO_VECTOR in \$IDBDEF. These changes allow a driver for devices in a combo device to find the beginning of the CSRs and Vector table for the combo device.

V03-050 RLRKDZ1 Robert L. Rappaport 9-Dec-1983
Add ability to get to BIIC registers thru \$KDZDEF so that a node can read its own BIIC registers without having to know what node it is.

V03-049 RSH0087 R. Scott Hanna 07-Dec-1983
Move \$KGBDEF to STARDEFFL.SDL

V03-048 RLRKDZ Robert L. Rappaport 6-Dec-1983
Add \$KDZDEF, virtual memory offsets to internal KDZ11 registers and devices.

V03-047 ACG0377 Andrew C. Goldstein, 6-Dec-1983 11:46
Realign JIB fields after WMC0020

V03-046 SRB0106 Steve Beckhardt 18-Nov-1983
Added EPID field to LKB, reduced LKB\$L_REFCNT field to a word.

V03-045 RLRI08SS Robert L. Rappaport 8-Nov-1983
Add \$I08SSDEF, Scorpio I/O space layout.

V03-044 KPL0002 Peter Lieberwirth 17-Oct-1983
Add IMP\$V_RUH_SYNCH to \$IMPDEF.

V03-043 SSA00002 Stan Amway 30-Sep-1983
Add IPL\$_PERFMON to \$IPLDEF.

V03-042 TMK0002 Todd M. Katz 18-Aug-1983
Add LNMTH\$V_GROUP and LNMTH\$V_SYSTEM to \$LNMSTRDEF.

V03-041 KDM0068 Kathleen D. Morse 5-Aug-1983
Add \$IOUV1DEF.

V03-040 RSH0047 R. Scott Hanna 24-Jul-1983
Add environmental ID definitions to \$KGBDEF

V03-039 RNG0039 Rod N. Gamache 22-Jul-1983
Add new DECnet FAST Interface definitions \$FFIDEF.

V03-038 LJK0217 Lawrence J. Kenah 26-Jun-1983
Put IHD\$V_DBGDMT back into \$IHDDEF

V03-037 LJK0214 Lawrence J. Kenah 24-Jun-1983
Remove Image Control Block (nee ICB) until various naming
issues are resolved.

V03-036 RPG0036 Bob Grosso 24-Jun-1983
Add new IHDDEF and ISDOLDDEF.

V03-035 CDS0001 Christian D. Saether 23-Jun-1983
Add FCBSL_ACCLKID field.

V03-034 RPG0034 Bob Grosso 23-Jun-1983
Add new Known File structures, KFRH, KFE, KFD, KFPB,
as well as ICB which is for the image activator.

V03-033 RNG0033 Rod Gamache 21-Jun-1983
Added IRP\$Q_STATION to overlay IRP\$Q_NT_PRVMSK.

V03-032 SRB0095 Steve Beckhardt 21-Jun-1983
Added LKB\$M_PROTECT status bit

V03-031 ROW0185 Ralph O. Weber 21-JUN-1983
Add block transfer fields to the connection manager CDRP
extension to IRP definition.

V03-030 LY0382 Larry Yetto 16-JUN-1983 14:16:40
Add IRP\$V_JNL_REMREQ to IRP STS flags

V03-029 SRB0091 Steve Beckhardt 1-Jun-1983
Removed several state codes from \$LKBDEF. Renamed other
state codes.

V03-028 SRB0087 Steve Beckhardt 24-May-1983
Added RESEND status bit to \$LKBDEF

V03-027 SRB0082 Steve Beckhardt 28-Apr-1983
Removed message queue from CDRP in \$IRPDEF

V03-026 TMK0001 Todd M. Katz 14-APR-1983
Make several changes to \$LNMSTRDEF. Delete LNMTH\$LOGNAM,
replace LNMTH\$V_SUBTABLE with LNMTH\$V_DIRECTORY, and add
LNMBS\$V_NODELETE.

V03-025 ROW0181 Ralph O. Weber 14-APR-1983
Add IRP\$VAL7 and IRP\$VAL8. Eventually, these fields
should replace currently used fields. However, the current
fields cannot be deleted yet. Therefore, the CDRP and the
IRP will be bigger than we want for a few weeks.

V03-024 MMD0138 Meg Dumont, 13-Apr-1983 10:17
Add HD4DEF which will contain extension to HDR1 FILE IDENTIFIER
field on magnetic tape

V03-023 SRB0075 Steve Beckhardt 6-Apr-1983
More changes to \$LKBDEF.

V03-022 SRB0072 Steve Beckhardt 25-Mar-1983
Added some new definitions in \$LKBDEF.

V03-021 STJ3073 Steven T. Jeffreys 25-Mar-1983
- Added FCB\$HIGHWATER
- Added FCB\$V_ERASE

V03-020 WMC0020 Wayne Cardoza 15-Mar-1983
Add MAXJOBS and MAXDETACH to JIB

V03-019 SRB0069 Steve Beckhardt 9-Mar-1983
Added NOQUOTA status bit to LKBDEF.

V03-018 KTA3037 Kerbey T. Altmann 11-Feb-1983
Add FLOAT definition to IO750DEF.

V03-017 MIR0022 Michael I. Rosenblum 19-Jan-1983
Add terminal speific IDB definition.

V03-016 ROW0156 Ralph O. Weber 11-JAN-1983
Add connection manager extension to CDRP portion of the IRP.
Remove hard coded filler offsets in IRP to be symbolic.
Reorder connection manager extension to CDRP so that the VAL1
through VAL6 fields overlay the fields in the block transfer
CDRP extension.

V03-015 WMC0015 Wayne Cardoza 9-Jan-1982
Add back KFPDEF which was accidentally removed in V03-012.

V03-014 ACG0307 Andrew C. Goldstein, 7-Jan-1983 16:30
Remove privilege mask and UIC from JIB

V03-013 SRB0060 Steve Beckhardt 7-Jan-1983
Added more definitions to \$LKBDEF

V03-012 ACG0307 Andrew C. Goldstein, 30-Dec-1982 17:42
Add rights database definitions (\$KGBDEF)

V03-011 SRB0057 Steve Beckhardt 15-Dec-1982
Reordered fields and added new fields and definitions
in \$LKBDEF for distributed lock manager.

V03-010 SRB0056 Steve Beckhardt 14-Dec-1982
Changed IPL\$ SYNCH and IPL\$ TIMER to be 8 (instead of 7).
Added IPL\$_TIMERFORK equal to 7.

V03-009 JMT0073 Jim Teague 09-Dec-1982
Add \$IHSDEF fields IHSSL_DMTVBN and IHSSL_DMTBYTES for
description of Debugger module/psect information.
Also define IHDSV_DBGDMT bit in IHDSL_LNKFLAGS in \$IHDDEF
to indicate presence of above fields.

V03-008 ACG0303 Andrew C. Goldstein, 9-Dec-1982 15:12

Add FILL attribute to extraneous names

V03-007 DMW4013 DMWalp 1-Dec-1982
Added \$LNMSTRDEF

V03-006 JWH0130 Jeffrey W. Horn 19-Nov-1982
Change IMP\$C_NPIOFILES to 63 so that the Process Perm
IFB/IRB tables take up a full page.

V03-005 KPL0001 Peter Lieberwirth 13-Oct-1982
Add IMP\$V_RECOVERY to \$IMPDEF

V03-004 TCM0003 Trudy C. Matthews 11-Aug-1982
Add IO790\$AL_PERABS to \$IO790DEF.

V03-003 JWH0001 Jeffrey W. Horn 29-Jul-1982
Add IMP\$V_RUH to \$IMPDEF.

V03-002 TCM0002 Trudy C. Matthews 28-Jul-1982 09:45
Change \$IO7VVDEF to \$IO790DEF. Remove 11/790-specific
definitions (\$PAMMDEF and SBIA register definitions) to
11/790-specific definition file, [SYSLOA.SRC]790DEF.MDL.

V03-001 LMP0036 L. Mark Pilant, 28-Jun-1982 13:37
Add space in the FCB definition for the Access Control List
queue listhead.

```
module $F11BDEF;
/**
/* F11B - System wide F11BXQP structures.
/*
/* This structure is actually part of the XQP impure area and
/* is pointed to by CTL$GL_F11BXQP. That cell is initialized
/* during process creation by the XQP initialization code.
/*
/*-
aggregate F11BDEF structure prefix F11B$:
  XQPQUEUE quadword;      /* XQP per-process queue.
  DISPATCH longword unsigned; /* Address of XQP dispatch routine.
  CODESIZE longword unsigned; /* Size of XQP code in bytes.
  CODEBASE longword unsigned; /* Base address of XQP code.
  IMPSIZE longword unsigned; /* Size of impure area in bytes.
  IMPBASE longword unsigned; /* Base address of XQP impure area.
end F11BDEF;
end_module $F11BDEF;
```



```
module $F11BCDEF;
```

```
/*+
/* F11BC - Files 11 Block Cache
/*
/* Header area which describes block cache used by F11BXQP.
/*
/*-
aggregate F11BCDEF structure prefix F11BC$:
  BUFBASE longword unsigned; /* Base address of buffer area.
  BUFSIZE longword unsigned; /* Size of buffer area in bytes.

  SIZE word unsigned; /* Standard size field.
  TYPE byte unsigned; /* Standard type field.
  SUBTYPE byte unsigned; /* Standard subtype field.
  REALSIZE longword unsigned; /* Structure size as a longword.

  LBNHSHBAS longword unsigned; /* Base of LBN hash table.
  LBNHSHCNT word unsigned; /* Count of entries in LBN hash tbl.
  BFRCNT word unsigned; /* Total buffer count.

  BFRDBAS longword unsigned; /* Buffer descriptor base address.
  BFRLDBAS longword unsigned; /* Buffer lock descriptor base addr.

  BLHSHBAS longword unsigned; /* Base addr of buffer lock hash tbl.
  BLHSHCNT word unsigned; /* Num entries in buff lock hash tbl.
  FREEBFRL word unsigned; /* First free buffer lock block.

  constant NUM_POOLS equals 4; /* Number of buffer pools.

  POOL_LRU quadword dimension F11BC$K_NUM_POOLS; /* Per pool LRU listhead.
  POOL_WAITQ quadword dimension F11BC$K_NUM_POOLS; /* Per pool cache wait listhead.
  POOLAVAIL longword dimension F11BC$K_NUM_POOLS; /* Available buffers per pool.
  POOLCNT word dimension F11BC$K_NUM_POOLS; /* Count of buffers per pool.

  AMBIGQFL longword unsigned; /* Ambiguity queue forward link.
  AMBIGQBL longword unsigned; /* Ambiguity queue back link.

/*
/* Cache performance counters.
/*
  PROCESS_HITS longword unsigned; /* In-process buffer hits.
  VALID_HITS longword unsigned; /* Valid buffer cache hits.
  INVALID_HITS longword unsigned; /* Buffer found but invalid contents.
  MISSES longword unsigned; /* Buffer not in cache at all.
  DISK_READS longword unsigned; /* Buffer reads from disk.
  DISK_WRITES longword unsigned; /* Buffer writes to disk.
  CACHE_SERIAL longword unsigned; /* Cache serialization calls.
  CACHE_STALLS longword unsigned; /* Cache serialization stalls.
  BUFFER_STALLS longword unsigned; /* Stalls for lack of buffers.

  CACHENAME character length 24; /* Name of this cache (display only).
```

end F11BCDEF;

/*
/* Buffer descriptors.
/*

aggregate BFRDDEF structure prefix BFRDS;
 QFL longword unsigned; /* Queue forward link.
 QBL longword unsigned; /* Queue back link.

 LBN longword unsigned; /* LBN of buffer.
 UCB longword unsigned; /* UCB of buffer.

 LOCKBASIS longword unsigned; /* Unique file identifier.
 SEQNUM longword unsigned; /* Buffer validation sequence number.

 FLAGS_OVERLAY union fill;
 FLAGS byte unsigned; /* Status flags.
 FLAGS_BITS structure fill;
 POOL bitfield length 2; /* Pool number of this buffer.
 DIRTY bitfield mask; /* Buffer has been modified.
 VALID bitfield mask; /* Buffer has been read from disk.
 end FLAGS_BITS;
 end FLAGS_OVERLAY;
 BTYP byte unsigned; /* Buffer type.
 BFRL word unsigned; /* Index to buffer lock.
 CURPID word unsigned; /* Index of current process.
 NXTBFRD word unsigned; /* Index of next BFRD (hash chain).

end BFRDDEF;

/*
/* Buffer lock descriptor blocks.
/*

aggregate BFRLDEF structure prefix BFRLS;
 NXTBFRL word unsigned; /* Index to next BFRL in list.
 REFCNT word unsigned; /* Number of buffers backed by this lock.
 LKID longword unsigned; /* Lock ID of buffer lock.

 LCKBASIS longword unsigned; /* Unique file identifier.
 PARLKID longword unsigned; /* Unique volume set identifier.

end BFRLDEF;

end_module \$F11BCDEF;


```
module $FCBDEF;
```

```
/*+
```

```
/* FCB - FILE CONTROL BLOCK
```

```
/*
```

```
/* THERE IS ONE FILE CONTROL BLOCK FOR EACH UNIQUELY ACCESSED FILE ON A
```

```
/* VOLUME. THE FILE CONTROL BLOCK PROVIDES THE VEHICLE WHEREBY SHARED
```

```
/* ACCESS TO A FILE MAY BE CONTROLLED.
```

```
/*-
```

```
aggregate FCBDEF structure prefix FCB$;
```

```
FCBFL longword unsigned;
```

```
FCBBL longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
ACCLKMODE byte unsigned;
```

```
EXFCB longword unsigned;
```

```
WLFL longword unsigned;
```

```
WLBL longword unsigned;
```

```
REFCNT word unsigned;
```

```
ACNT word unsigned;
```

```
WCNT word unsigned;
```

```
LCNT word unsigned;
```

```
TCNT word unsigned;
```

```
STATUS_OVERLAY union fill;
```

```
STATUS word unsigned;
```

```
STATUS BITS structure fill;
```

```
DIR bitfield;
```

```
MARKDEL bitfield;
```

```
BADBLK bitfield;
```

```
EXCL bitfield;
```

```
SPOOL bitfield;
```

```
RMSLOCK bitfield;
```

```
ERASE bitfield;
```

```
BADACL bitfield;
```

```
STALE bitfield;
```

```
DELAYTRNC bitfield;
```

```
end STATUS BITS;
```

```
end STATUS_OVERLAY;
```

```
FID_OVERLAY union fill;
```

```
FID word unsigned dimension 3;
```

```
FID_FIELDS structure fill;
```

```
FID_NUM word unsigned;
```

```
FID_SEQ word unsigned;
```

```
FID_RVN_OVERLAY union fill;
```

```
FID_RVN word unsigned;
```

```
FID_RVN_FIELDS structure fill;
```

```
FID_RVN byte unsigned;
```

```
FID_NMX byte unsigned;
```

```
end FID_RVN_FIELDS;
```

```
end FID_RVN_OVERLAY;
```

```
end FID_FIELDS;
```

```
end FID_OVERLAY;
```

```
SEGN word unsigned;
```

```
STVBN longword unsigned;
```

```
STLBN longword unsigned;
```

```
/* FCB LIST FORWARD LINK
```

```
/* FCB LIST BACKWARD LINK
```

```
/* SIZE OF FCB IN BYTES
```

```
/* STRUCTURE TYPE OF FCB
```

```
/* Access lock mode.
```

```
/* ADDRESS OF EXTENSION FCB
```

```
/* WINDOW LISTHEAD FORWARD LINK
```

```
/* WINDOW LISTHEAD BACKWARD LINK
```

```
/* Total references to this FCB.
```

```
/* FILE ACCESS COUNT
```

```
/* FILE WRITER COUNT
```

```
/* FILE LOCK COUNT
```

```
/* COUNT OF TRUNCATE LOCKS
```

```
/* FILE STATUS
```

```
/* FCB IS A DIRECTORY LRU ENTRY
```

```
/* FILE IS MARKED FOR DELETE
```

```
/* BAD BLOCK ENCOUNTERED IN FILE
```

```
/* FILE IS EXCLUSIVELY ACCESSED
```

```
/* FILE IS AN INTERMEDIATE SPOOL FILE
```

```
/* FILE IS OPEN WITH RMS RECORD LOCKING
```

```
/* ERASE DATA WHEN BLOCKS REMOVED FROM FILE
```

```
/* ACL IS CORRUPT
```

```
/* Reconstruct FCB from header.
```

```
/* Delay truncation.
```

```
/* FILE IDENTIFICATION
```

```
/* FILE NUMBER
```

```
/* FILE SEQUENCE NUMBER
```

```
/* RELATIVE VOLUME NUMBER
```

```
/* SHORT FORM RVN
```

```
/* EXTENDED FILE NUMBER
```

```
/* FILE SEGMENT NUMBER
```

```
/* STARTING VIRTUAL BLOCK NUMBER
```

```
/* STARTING LOGICAL BLOCK NUMBER
```

```

HDLBN longword unsigned; /* LBN OF FILE HEADER
FILESIZE longword unsigned; /* FILE SIZE IN BLOCKS
EFBLK longword unsigned; /* END OF FILE VBN
VERSIONS word unsigned; /* MAXIMUM NUMBER OF VERSIONS IN DIRECTORY
DIRSEQ word unsigned; /* DIRECTORY USE SEQUENCE NUMBER
HIGHWATER longword unsigned; /* HIGH WATER MARK IN FILE
ACCLKID longword unsigned; /* Access lock ID.
LOCKBASIS longword unsigned; /* Lock basis for this FCB.
TRUNCVBN longword unsigned; /* VBN for delayed truncation.
CACHELKID longword unsigned; /* Cache interlock lock ID
ORB structure; /* Object's Rights Block
  FILEOWNER structure longword unsigned; /* FILE OWNER UIC
    UICMEMBER word unsigned; /* MEMBER NUMBER
    UICGROUP word unsigned; /* GROUP NUMBER
  end FILEOWNER;
  FILL_5 longword unsigned fill; /* ACL mutex
  FILL_3 longword unsigned fill; /* Structure size & type
  FILL_6 longword unsigned fill; /* Spare + ref count
  ACMODE quadword unsigned; /* Access mode protection vector
  SYS_PROT structure longword unsigned; /* Protection word/vector
    FILEPROT word unsigned; /* FILE PROTECTION MASK
    FILL_4 word unsigned fill;
  end SYS_PROT;
  OWN_PROT longword unsigned; /* Owner protection
  GRP_PROT longword unsigned; /* Group protection
  WOR_PROT longword unsigned; /* World protection
  ACLFL longword unsigned; /* ACCESS CONTROL LIST FORWARD LINK
  ACLBL longword unsigned; /* ACCESS CONTROL LIST BACKWARD LINK
  MIN_CLASS_PROT structure; /* Minimum security classification mask
    FILL_2 byte dimension 20 fill; /* see structure $CLSDEF
  end MIN_CLASS_PROT;
  MAX_CLASS_PROT structure; /* Maximum security classification mask
    FILL_2 byte dimension 20 fill; /* see structure $CLSDEF
  end MAX_CLASS_PROT;
end ORB;
DIRINDX longword unsigned; /* Directory index pointer
constant 'LENGTH' equals . prefix FCB$ tag K; /* LENGTH OF STANDARD FCB
constant 'LENGTH' equals . prefix FCB$ tag C; /* LENGTH OF STANDARD FCB
end FCBDEF;

end_module $FCBDEF;

```



```
module $FFIDEF;
/**
/* FFI - DECnet-VAX Fast Interface
/*
/*-

aggregate FFIDEF structure prefix FFIS;
  FL longword unsigned; /*FORWARD QUEUE LINK
  BL longword unsigned; /*BACKWARD QUEUE LINK
  SIZE word unsigned; /*BLOCK SIZE
  TYPE byte unsigned; /*BLOCK TYPE
  SPARE byte unsigned; /*DATALINK SPARE BYTE
  CTX_DL longword unsigned; /*DATALINK CONTEXT AREA
  XMIT longword unsigned; /*DATALINK TRANSMIT ROUTINE ADDRESS
  XMIT_DONE longword unsigned; /*CALLER'S TRANSMIT DONE ROUTINE ADDRESS
  RECV_DONE longword unsigned; /*CALLER'S RECEIVE DONE ROUTINE ADDRESS
  ERROR longword unsigned; /*CALLER'S ERROR HANDLER
  SHUT_DONE longword unsigned; /*CALLER'S SHUTDOWN COMPLETE ROUTINE ADDRESS
  SPARE0 longword unsigned; /* spare
  SPARE1 longword unsigned; /* spare
  SPARE2 longword unsigned; /* spare
  SPARE3 longword unsigned; /* spare
  DL_UCB longword unsigned; /*DATALINK UCB ADDRESS
  PID longword unsigned; /*CALLER'S PID (or zero)
  CHAN word unsigned; /*CALLER'S CHAN (or zero)
  constant CTX_USER equals . prefix FFIS tag G; /*CALLER'S CONTEXT BLOCK
  constant 'LENGTH' equals . prefix FFIS tag K; /*LENGTH OF A STANDARD FFI
  constant 'LENGTH' equals . prefix FFIS tag C; /*LENGTH OF A STANDARD FFI
end FFIDEF;

end_module $FFIDEF;
```

```
module $FKBDEF;
```

```
/*
```

```
/* FKB - FORK BLOCK
```

```
/*
```

```
/* A FORK BLOCK DESCRIBES THE CONTEXT OF A FORK PROCESS. EACH UNIT CONTROL
```

```
/* BLOCK CONTAINS A FORK BLOCK AS ITS FIRST SIX LONGWORDS.
```

```
/*-
```

```
aggregate FKBDEF structure prefix FKB$;
```

```
  FQFL longword unsigned;
```

```
  FQBL longword unsigned;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  FIPL byte unsigned;
```

```
  FPC longword unsigned;
```

```
  FR3 longword unsigned;
```

```
  FR4 longword unsigned;
```

```
  constant 'LENGTH' equals . prefix FKB$ tag K;
```

```
  constant 'LENGTH' equals . prefix FKB$ tag C;
```

```
/*FORK QUEUE FORWARD LINK
```

```
/*FORK QUEUE BACKWARD LINK
```

```
/*SIZE OF FKB IN BYTES
```

```
/*STRUCTURE TYPE OF FKB
```

```
/*FORK INTERRUPT PRIORITY LEVEL
```

```
/*FORK PC
```

```
/*FORK R3
```

```
/*FORK R4
```

```
/*STANDARD LENGTH OF FKB
```

```
/*STANDARD LENGTH OF FKB
```

```
end FKBDEF;
```

```
end_module $FKBDEF;
```



```
module $GSDDEF;
```

```
/**
/* GLOBAL SECTION DESCRIPTOR BLOCK
/*-
```

```
aggregate GSDDEF structure prefix GSD$;
```

```
  GSDFL OVERLAY union fill;
```

```
    GSDFL longword unsigned;
```

```
    GSDFL BITS structure fill;
```

```
      VALID bitfield mask;
```

```
      LOCKED bitfield mask;
```

```
      DELPEND bitfield mask;
```

```
      INITFAIL bitfield mask;
```

```
      DUPGSD bitfield mask;
```

```
    end GSDFL BITS;
```

```
  end GSDFL_OVERLAY;
```

```
  GSDBL longword unsigned;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  HASH byte unsigned;
```

```
  PCBUIC OVERLAY union fill;
```

```
    PCBUIC longword unsigned;
```

```
    PCBUIC FIELDS structure fill;
```

```
      FILL 6 byte dimension 2 fill prefix GSDDEF tag $$;
```

```
      PCBGRP word unsigned;
```

```
    end PCBUIC FIELDS;
```

```
  end PCBUIC_OVERLAY;
```

```
  FILUIC longword unsigned;
```

```
  PROT word unsigned;
```

```
  GSTX word unsigned;
```

```
  "IDENT" longword unsigned;
```

```
  ORB longword unsigned;
```

```
  FLAGS word unsigned;
```

```
  GSDNAM character;
```

```
  constant "LENGTH" equals . prefix GSD$ tag K;
```

```
  constant "LENGTH" equals . prefix GSD$ tag C;
```

```
  #GSDLEN = . ;
```

```
/*POINTER TO NEXT GSD
```

```
/*SH MEM GSD FLAG, SET IF VALID ENTRY
```

```
/*SH MEM GSD FLAG, SET IF ENTRY LOCKED
```

```
/*SH MEM GSD FLAG, GS MARKED FOR DELETE
```

```
/*SH MEM GSD FLAG, SET WHEN GS INIT FAILS
```

```
/*SH MEM GSD FLAG, DUPLICATE GSD FOUND
```

```
/*POINTER TO PREVIOUS GSD
```

```
/*SIZE OF GSD IN BYTES
```

```
/*STRUCTURE TYPE CODE FOR GSD
```

```
/*HASH FOR GSD NAME
```

```
/*UIC OF CREATOR OF SECTION, FROM HIS PCB
```

```
/*GROUP OF CREATOR OF SECTION, FROM PCB
```

```
/*OWNER OF FILE, UIC FROM FCB
```

```
/*PROTECTION MASK
```

```
/*GLOBAL SECTION TABLE INDEX
```

```
/*IDENTIFICATION OF GLOBAL SECTION
```

```
/*OBJECT RIGHTS BLOCK LOCATOR
```

```
/*SECTION FLAGS
```

```
/*LOCAL MEMORY AND SHARED MEMORY SECTION NAME
```

```
/*LENGTH OF LOCAL MEMORY GSD
```

```
/*LENGTH OF LOCAL MEMORY GSD
```

```
/*
/* THE FOLLOWING FIELDS ARE ONLY FOUND IN EXTENDED GSD'S. THESE ARE USED
/* WHENEVER A GSD IS NEEDED WITHOUT A SECTION TABLE ENTRY, I.E., FOR SHARED
/* MEMORY AND FOR PAGES MAPPED BY PFN.
/*
```

```
  FILL 2 byte fill prefix GSDDEF tag $$;
```

```
  BASEPFN longword unsigned;
```

```
  PAGES longword unsigned;
```

```
  REFCNT longword unsigned;
```

```
  PFNGSDNAM character;
```

```
  constant EXTGSDLNG equals . prefix GSD$ tag K;
```

```
  constant EXTGSDLNG equals . prefix GSD$ tag C;
```

```
/*SPARE BYTE
```

```
/*FIRST RELATIVE BASE PFN
```

```
/*COUNT OF PAGES AT FIRST BASE PFN
```

```
/*FIRST PROCESSOR PTE REF COUNT
```

```
/*PFN-MAPPED SECTION NAME
```

```
/*MINIMUM EXTENDED GSD LENGTH
```

```
/*MINIMUM EXTENDED GSD LENGTH
```

```
/*
/* THE FOLLOWING FIELDS ARE CONTAINED ONLY IN SHARED MEMORY GSD'S. THE LENGTH,
/* GSD$C_SHMGSDLNG, IS ONLY THE CONSTANT SIZE OF THE GSD. IN ADDITION, THERE IS
```



```
/* ONE LONGWORD FOR EACH PROCESSOR AND TWO LONGWORDS FOR EACH BASE PFN-SIZE PAIR.
```

```
/*
```

```
end GSDDEF;
```

```
aggregate GSDDEF1 structure prefix GSD$;
```

```

  FILL_7 byte dimension #GSDLEN+45 fill prefix GSDDEF tag $$; /*SHMEM GLOBAL SECTION NAME
  LOCK byte unsigned; /*INTERPROCESSOR LOCK FOR GSD
  PROCCNT byte unsigned; /*NUMBER OF PROCESSOR REF. COUNTS IN GSD
  CREATPORT byte unsigned; /*PORT ! FOR CREATOR PROCESSOR
  DELETPORT byte unsigned; /*PORT ! FOR DELETOR PROCESSOR
  constant PFNBASMAX equals 4 prefix GSD tag $C; /*MAXIMUM ! OF PFN BASES ALLOWED
  BASPFN1 longword unsigned; /*FIRST BASE PFN FOR SECTION PAGES
  BASCNT1 longword unsigned; /*CNT OF SECTION PAGES AT FIRST BASE PFN
  /*FIRST PFN/PAGE COUNT PAIR
  /* NOTE: THE NUMBER OF QUADWORDS
  /* RESERVED FOR PFN/PAGE COUNTS PAIRS
  /* MUST BE EQUAL TO GSD$C_PFNBASMAX-1.
  FILL_3 quadword fill prefix GSDDEF tag $$; /*LENGTH OF CONSTANT PART OF SHM GSD
  FILL_4 quadword fill prefix GSDDEF tag $$; /*LENGTH OF CONSTANT PART OF SHM GSD
  FILL_5 quadword fill prefix GSDDEF tag $$; /*PTE COUNT FOR FIRST PROCESSOR
  constant SHMGSDLNG equals . prefix GSD$ tag K;
  constant SHMGSDLNG equals . prefix GSD$ tag C;
  PTECNT1 longword unsigned;

```

```
end GSDDEF1;
```

```
end_module $GSDDEF;
```



```
module $HD1DEF;
```

```
/*+
/* HDR1 ANDS1 MAGNETIC TAPE LABEL
/* THIS IS THE FIRST LABEL IN THE FILE LABEL HEADER SET. IF IDENTIFIES THE FILE.
/*-
```

```
aggregate HD1DEF structure prefix HD1$;
```

HD1LID longword unsigned;	/*LABEL IDENTIFIER AND NUMBER 'HDR1'
FILEID character length 17;	/*FILE IDENTIFIER
FILESETID character length 6;	/*FILE SET IDENTIFIER
FILESECNO character length 4;	/*FILE SECTION NUMBER
FILESEQNO character length 4;	/*FILE SEQUENCE NUMBER
GENNO character length 4;	/*FILE GENERATION NUMBER
GENVER character length 2;	/*FILE GENERATION VERSION NUMBER
CREATEDT character length 6;	/*CREATION DATE (YYDDD)
EXPIREDT character length 6;	/*EXPIRATION DATE
FILACCESS byte unsigned;	/*FILE ACCESS
BLOCKCNT character length 6;	/*BLOCK COUNT
SYSCODE character length 13;	/*SYSTEM CODE
FILL 1 character length 7 fill prefix HD1DEF tag \$\$;	/*SPACES

```
end HD1DEF;
```

```
end_module $HD1DEF;
```



```
module $HD2DEF;
```

```
/**
/* HDR2 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE SECOND LABEL IN FILE LABEL HEADER SET.
/* THE FILE ATTRIBUTES HAVE BEEN REMOVED FROM HDR2, AND PLACED IN HDR3.
/* THE FIELDS REMAIN IN THE DEFINITION TO SUPPORT OLD TAPES.
/*-
```

```
aggregate HD2DEF structure prefix HD2$;
```

HD2LID longword unsigned;	/*LABEL IDENTIFIER AND NUMBER 'HDR2'
RECFORMAT byte unsigned;	/*RECORD FORMAT
BLOCKLEN character length 5;	/*BLOCK LENGTH
RECLN character length 5;	/*RECORD LENGTH
RECATR1 character length 20;	/*FIRST 20 BYTES OF FILES-11 RECORD ATTRIBUTES
FILL 1 character fill prefix HD2DEF tag \$\$;	/*SPACES
FORMCNTRL byte unsigned;	/*FORMS CONTROL
RECATR2 character length 12;	/*LAST 12 BYTES OF FILES-11 RECORD ATTRIBUTES
FILL 2 character fill prefix HD2DEF tag \$\$;	/*SPACES
BUFOFF character length 2;	/*BUFFER OFFSET
FILL 3 character length 20 fill prefix HD2DEF tag \$\$;	/*SPACES

```
end HD2DEF;
```

```
end_module $HD2DEF;
```



```
module $HD3DEF;
```

```
/**
/* HDR3 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE THIRD LABEL IN FILE LABEL HEADER SET.
/* IT IDENTIFIES THE FILE ATTRIBUTES.
/*-
```

```
aggregate HD3DEF structure prefix HD3$;
```

```
    HD3LID longword unsigned;          /*LABEL IDENTIFIES AND NUMBER 'HDR3'
    RECATR character length 64;        /*64 BYTES OF FILES-11 RECORD ATTRIBUTES
    FILL 1 character length 12 fill prefix HD3DEF tag $$; /*SPACES
end HD3DEF;
```

```
end_module $HD3DEF;
```

```
module $HD4DEF;
```

```
/**
/* HDR4 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE FOURTH LABEL IN FILE LABEL HEADER SET.
/* IT CONTAINS THE LONG FILENAME EXTENSION TO THE HDR1 FILE IDENTIFIER
/* FOR VMS LONG FILE NAMES
/*-
```

```
aggregate HD4DEF structure prefix HD4$;
```

```
    HD4LID longword unsigned;          /*LABEL IDENTIFIER AND NUMBER 'HDR4'
    FILEID_EXT_SIZE byte unsigned;      /*SIZE OF FILE ID EXT FOR ANSI 4 VOLUMES
    FILEID_EXT character length 62;     /*EXTENSION OF HDR1 FILEID
    FILEID_EXT_V3 character length 2;    /*SIZE OF FILE ID EXT FOR ANSI 3 VOLUMES
    FILL 1 character length 13 fill prefix HD4DEF tag $$; /*SPACES
```

```
end HD4DEF;
```

```
end_module $HD4DEF;
```



```
module $IAFDEF;
```

```
/**
/* IAF - IMAGE ACTIVATOR FIXUP SECTION
/*
/* THE IMAGE ACTIVATOR FIXUP SECTION IS AN IMAGE SECTION THAT IS CREATED
/* BY THE LINKER AND USED BY THE IMAGE ACTIVATOR TO MODIFY THE IMAGE AS
/* IT IS ACTIVATED. THIS IS DONE TO MAINTAIN THE POSITION INDEPENDENCE
/* OF EXTERNAL REFERENCES.
/*-
```

```
aggregate IAFDEF structure prefix IAF$;
```

```
  IAFLINK longword unsigned; /* Link for image activator use
  FIXUPLNK longword unsigned; /* Link for shareable image fixups
  SIZE word unsigned; /* Size of fixed part of IAF
  FLAGS OVERLAY union fill;
    FLAGS word unsigned; /* Flags
    FLAGS BITS structure fill; /* This is in a shareable image
      SHR bitfield;
    end FLAGS BITS;
  end FLAGS OVERLAY;
  G FIXOFF longword unsigned; /* Offset to g^ address data
  DOTADROFF longword unsigned; /* Offset to .address fixup data
  CHGPRTOFF longword unsigned; /* Offset to isect change prot. data
  SHLSTOFF longword unsigned; /* Offset to shareable image list
  SHRIMGCNT longword unsigned; /* Number of shareable images in shlst
  SHLEXTRA longword unsigned; /* Number of extra shareable images allowed
  PERMCTX longword unsigned; /* Permanent sharable image context
  FILL_1 longword fill prefix IAFDEF tag $$; /* Spare
  FILL_2 longword fill prefix IAFDEF tag $$; /* Spare
  FILL_3 longword fill prefix IAFDEF tag $$; /* Spare
  FILL_4 longword fill prefix IAFDEF tag $$; /* Spare
  FILL_5 longword fill prefix IAFDEF tag $$; /* Spare
  FILL_6 longword fill prefix IAFDEF tag $$; /* Spare
  constant "LENGTH" equals . prefix IAF$ tag K; /* Length of fixed area
  constant "LENGTH" equals . prefix IAF$ tag C; /* Length of fixed area
end IAFDEF;
```

```
end_module $IAFDEF;
```



```
module $ICPDEF;
```

```
/*+
```

```
/* ICP - CHANGE IMAGE SECTION PROTECTION DATA
```

```
/*
```

```
/* THIS STRUCTURE IS USED IN THE IMAGE FIXUP SECTION BY THE LINKER
```

```
/* TO INFORM THE IMAGE ACTIVATOR OF THE IMAGE SECTIONS THAT NEED
```

```
/* THEIR PROTECTION CHANGED.
```

```
/*-
```

```
aggregate ICPDEF structure prefix ICPS;
```

```
    BASEVA longword unsigned;
```

```
    NPAGES word unsigned;
```

```
    NEWPRT word unsigned;
```

```
    constant 'LENGTH' equals . prefix ICPS tag K;
```

```
    constant 'LENGTH' equals . prefix ICPS tag C;
```

```
end ICPDEF;
```

```
end_module $ICPDEF;
```

```
/* virtual address of start of section
```

```
/* number of pages to change protection on
```

```
/* new protection
```

```
/* size of one section's data
```

```
/* size of one section's data
```



```
module $IDBDEF;
```

```
/*+
/* IDB - INTERRUPT DISPATCH BLOCK
/*
/* AN INTERRUPT DISPATCH BLOCK PROVIDES THE INFORMATION NECESSARY FOR A
/* UNIT INDEPENDENT, BUT CONTROLLER SPECIFIC, INTERRUPT DISPATCHER TO
/* DISPATCH INTERRUPTS TO THE PROPER DRIVER TO HANDLE AN INTERRUPT ON
/* A DEVICE UNIT.
/*-
```

```
aggregate IDBDEF structure prefix IDB$;
```

```
  CSR longword unsigned;
  OWNER longword unsigned;
  SIZE word unsigned;
  TYPE byte unsigned;
  VECTOR byte unsigned;
  UNITS word unsigned;
  TT_ENABLE byte unsigned;
  COMBO_CSR_OFFSET byte;
  COMBO_VECTOR_OFFSET byte;
  SPARE1 byte;
  SPARE2 word;
  ADP longword unsigned;
  UCBLST longword unsigned dimension 8;
```

```
  constant 'LENGTH' equals . prefix IDB$ tag K;
  constant 'LENGTH' equals . prefix IDB$ tag C;
```

```
end IDBDEF;
```

```
end_module $IDBDEF;
```

```
/*CONTROLLER CSR ADDRESS
/*OWNER UCB ADDRESS
/*SIZE OF IDB IN BYTES
/*STRUCTURE TYPE OF IDB
/*CONTROLLER VECTOR OFFSET
/*NUMBER OF UNITS (SIZE OF UCBLST)
/* DZ32 line enable field
/*OFFSET TO START OF COMBO DEVICE STYLE CSRS
/*OFFSET TO START OF COMBO STYLE DEVICE VECTORS
/*A SPARE BYTE
/*A SPARE WORD
/*ADDRESS OF UBA ADAPTER CONTROL BLOCK
/*UCB OR SECONDARY IDB ADDRESSES
/*(DEFAULT OF 8)
/*LENGTH OF STANDARD IDB
/*LENGTH OF STANDARD IDB
```

```
module $IFDDEF;
```

```
/**
```

```
/* IMAGE FILE DESCRIPTOR BLOCK - RETURNED BY IMAGE ACTIVATOR
```

```
/**-
```

```
aggregate IFDDEF structure prefix IFD$;
```

```
  SIZE word unsigned;
```

```
  FILNAMOFF word unsigned;
```

```
  FILL_1 word fill prefix IFDDEF tag $$;
```

```
  FILL_2 word fill prefix IFDDEF tag $$;
```

```
  CHAN word unsigned;
```

```
  CMCHAN word unsigned;
```

```
  CMKFIADR longword unsigned;
```

```
  FLAGS OVERLAY union fill;
```

```
    FLAGS word unsigned;
```

```
    FLAGS BITS structure fill;
```

```
      EXEONLY bitfield mask;
```

```
      PRIV bitfield mask;
```

```
      SETVECTOR bitfield mask;
```

```
    end FLAGS BITS;
```

```
  end FLAGS_OVERLAY;
```

```
  FILL_3 word fill prefix IFDDEF tag $$;
```

```
  CURPROG quadword unsigned;
```

```
  constant 'LENGTH' equals . prefix IFD$ tag K;
```

```
  constant 'LENGTH' equals . prefix IFD$ tag C;
```

```
end IFDDEF;
```

```
end_module $IFDDEF;
```

```
/*SIZE IN BYTES OF IMAGE FILE DESCRIPTOR
```

```
/*OFFSET TO RESULTANT FILE NAME STRING
```

```
/*RESERVED OFFSET 1
```

```
/*RESERVED OFFSET 2
```

```
/*CHANNEL ON WHICH IMAGE FILE IS OPEN
```

```
/*COMPATIBILITY MODE CHANNEL
```

```
/*COMPATIBILITY MODE IMAGE
```

```
/*KNOWN FILE ENTRY ADDRESS OR 0
```

```
/*IMAGE FILE DESCRIPTOR FLAGS
```

```
/*EXECUTE ONLY FILE
```

```
/*IMAGE INSTALLED WITH ENHANCED PRIVILEGE
```

```
/*PRIVILEGED VECTORS TO BE INSTALLED
```

```
/*SPARE WORD
```

```
/*STRING DESCRIPTOR FOR CURRENTLY
```

```
/*RUNNING PROGRAM NAME
```

```
/*LENGTH OF FIXED AREA OF IFD
```

```
/*LENGTH OF FIXED AREA OF IFD
```



```
module $IHDDEF;
```

```
/**
/* IMAGE HEADER RECORD DEFINITIONS - FIRST RECORD OF IMAGE HEADER
/*-
```

```
aggregate IHDDEF structure prefix IHD$;
```

```
SIZE word unsigned;          /* Size in bytes of Image Header record
ACTIVOFF word unsigned;       /* Byte offset to activation data
SYMDBGOFF word unsigned;     /* Byte offset to symbol table and debug data
IMGIDOFF word unsigned;      /* Byte offset to image ident data
PATCHOFF word unsigned;     /* Byte offset to patch data
SPARE word unsigned;         /* spare
MAJORID word unsigned;       /* Major id
constant MAJORID equals '02' prefix IHD tag $K; /* Major id value
MINORID word unsigned;       /* Minor id
constant MINORID equals '05' prefix IHD tag $K; /* Minor id value
HDRBLKCNT byte unsigned;     /* Count of header blocks
IMGTYPE byte unsigned;       /* Image type
```

```
/*
/* IMAGE TYPE CODES
/*
```

```
constant EXE equals 1 prefix IHD tag $K; /* Executable image
constant LIM equals 2 prefix IHD tag $K; /* Linkable image

FILL 2 word fill prefix IHDDEF tag $$; /* Reserved
PRIVREQS quadword unsigned; /* Requested privilege mask
IOCHANCNT word unsigned; /*! of channels requested
/*0 if default
IMGIOCNT word unsigned; /*! of pages of image i/o section requested
/*0 if default

LNKFLAGS OVERLAY union;
LNKFLAGS longword unsigned; /* Linker produced image flags
LNKFLAGS BITS structure;
LNKDEBUG bitfield mask; /* Full debugging requested
LNKNOTFR bitfield mask; /* First transfer address missing
NOPOBUFS bitfield mask; /* RMS use of P0 for image i/o disabled
PICIMG bitfield mask; /* Image is position independent
POIMAGE bitfield mask; /* Image is in P0 space only
DBGDMT bitfield mask; /* Image header has dmt fields
INISHR bitfield mask; /* Transfer array contains valid IHASL INISHR
FILL 3 bitfield length 17 fill prefix IHDDEF tag $$; /* FILL OUT TO HIGH BYTE OF LONG WORD
MATCHCTL bitfield mask length 3; /* Match control for linkable image
end LNKFLAGS BITS;
end LNKFLAGS OVERLAY;
"IDENT" longword unsigned; /* GBL SEC ident value for linkable image
SYSVER longword unsigned; /* SYSSK_VERSION or 0 if not linked with exec
IAFVA longword unsigned; /* Relative virtual address of fixup info
constant "LENGTH" equals . prefix IHD$ tag K; /* Length of fixed area
constant "LENGTH" equals . prefix IHD$ tag C; /* Length of fixed area
```

```
SKIP character length 510 - .; /* ALIAS should be last word in 512 byte block
ALIAS word unsigned; /* Code to use secondary image name
/*****
/*
```

```
/* Define legal range of ALIAS constants. MINCODE must be equal to the
/* lowest value and MAXCODE must be equal to the highest value.
/*
constant MINCODE equals -1 prefix IHD$ tag C;      /* Low bound of ALIAS values
constant NATIVE equals -1 prefix IHD$ tag C;      /* Native mode image
constant RSX equals 0 prefix IHD$ tag C;          /* RSX image produced by TKB
constant BPA equals 1 prefix IHD$ tag C;          /* BASIC plus analog
constant ALIAS equals 2 prefix IHD$ tag C;        /* Last 126 bytes contains ASCII of image to activate
constant CLI equals 3 prefix IHD$ tag C;          /* Image is a CLI, run LOGINOUT
constant MAXCODE equals 3 prefix IHD$ tag C;      /* High bound of ALIAS values
/*
/*****
```

```
/*
/* Generation number returned by IMGSHR IMG$GET_IHD to SYSIMGACT.
/* These do not appear in the image header but are inferred from the
/* contents of the image header
/*
constant GEN_XLNKR equals 1 prefix IHD tag $C;    /* Cross linker
constant GEN_NATIVE equals 2 prefix IHD tag $C;  /* First native mode image header.
/* does not have LNKFLAGS, SYSVER and IAFVA fields
constant GEN_LNKFLG equals 3 prefix IHD tag $C;  /* Native with LNKFLAGS longword added
/* does not have SYSVER and IAFVA fields
constant GEN_SYSVER equals 4 prefix IHD tag $C;  /* Native with LNKFLAGS and SYSVER added
/* does not have IAFVA field
constant GEN_FIXUP equals 5 prefix IHD tag $C;   /* Version III image
/* contains LNKFLAGS, SYSVER, and IAFVA fields
constant GEN_NEWISD equals 6 prefix IHD tag $C;  /* ISD size field is a byte
```

end IHDDEF;

end_module \$IHDDEF;


```
module $IHPDEF;
```

```
/**  
/* IMAGE HEADER PATCH SECTION OFFSETS  
/*-
```

```
aggregate IHPDEF structure prefix IHP$;
```

```
ECO1 longword unsigned;  
ECO2 longword unsigned;  
ECO3 longword unsigned;  
ECO4 longword unsigned;  
RW_PATSI2 longword unsigned;  
RW_PATADR longword unsigned;  
RO_PATSI2 longword unsigned;  
RO_PATADR longword unsigned;  
PATCOMTXT longword unsigned;  
PATDATE quadword unsigned;  
constant 'LENGTH' equals . prefix IHP$ tag K;  
constant 'LENGTH' equals . prefix IHP$ tag C;
```

```
/*DEC ECO LEVELS 1-32  
/*DEC ECO LEVELS 33-64  
/*DEC ECO LEVELS 65-98  
/*USER ECO LEVELS 99-132  
/*SIZE OF FREE RW PATCH AREA  
/*VIR ADDR OF NEXT FREE RW PATCH AREA  
/*SIZE OF FREE RO PATCH AREA  
/*VIR ADDR OF NEXT FREE RO PATCH AREA  
/*PATCH COMMAND TEXT VIRTUAL BLOCK NUMBER  
/*DATE OF MOST RECENT PATCH  
/*LENGTH OF PATCH HEADER SECTION  
/*LENGTH OF PATCH HEADER SECTION
```

```
end IHPDEF;
```

```
end_module $IHPDEF;
```



```
module $IHSDEF;
```

```
/**  
/* IMAGE HEADER SYMBOL TABLE AND DEBUG SECTION OFFSETS  
/*-
```

```
aggregate IHSDEF structure prefix IHSS;
```

```
    DSTVBN longword unsigned;
```

```
    GSTVBN longword unsigned;
```

```
    DSTBLKS word unsigned;
```

```
    GSTRECS word unsigned;
```

```
    DMTVBN longword unsigned;
```

```
    DMTBYTES longword unsigned;
```

```
    constant "LENGTH" equals . prefix IHSS tag K;
```

```
    constant "LENGTH" equals . prefix IHSS tag C;
```

```
/*DEBUG SYMBOL TABLE VIRTUAL BLOCK NUMBER
```

```
/*GLOBAL SYMBOL TABLE VIRTUAL BLOCK NUMBER
```

```
/*DEBUG SYMBOL TABLE BLOCK COUNT
```

```
/*GLOBAL SYMBOL TABLE RECORD COUNT
```

```
/*VBN OF DMT INFORMATION
```

```
/*LENGTH OF DMT INFO
```

```
/*LENGTH OF SYMBOL TABLE SECTION
```

```
/*LENGTH OF SYMBOL TABLE SECTION
```

```
end IHSDEF;
```

```
end_module $IHSDEF;
```



```
module $IHIDEF;
```

```
/*+  
/* IMAGE HEADER IDENTIFICATION SECTION OFFSETS  
/*-
```

```
aggregate IHIDEF structure prefix IHIS;
```

```
  IMGNAM character length 40;
```

```
  IMGID character length 16;
```

```
  LINKTIME quadword unsigned;
```

```
  LINKID character length 16;
```

```
  constant 'LENGTH' equals . prefix IHIS tag K;
```

```
  constant 'LENGTH' equals . prefix IHIS tag C;
```

```
end IHIDEF;
```

```
end_module $IHIDEF;
```

```
/*IMAGE NAME STRING
```

```
/*IMAGE IDENT STRING
```

```
/*DATE AND TIME THIS IMAGE WAS LINKED
```

```
/*STANDARD SYSTEM QUADWORD FORMAT
```

```
/*LINKER IDENT STRING
```

```
/*LENGTH OF IMAGE HEADER IDENT SECTION
```

```
/*LENGTH OF IMAGE HEADER IDENT SECTION
```



```
module $IHXDEF;
```

```
/*+
/* IMAGE HEADER RECORD DEFINITIONS - CROSS LINKER - MAJORID = "01"
/*      1ST RECORD OF IMAGE HEADER BLOCK
/*-
```

```
aggregate IHXDEF structure prefix IHX$;
```

```
    SIZE word unsigned;          /*SIZE IN BYTE OF IMAGE HEADER RECORD
    HDRBLKCNT byte unsigned;      /*COUNT OF BLOCKS IN IMAGE HEADER
    FILL_1 byte fill prefix IHXDEF tag $$; /*SPARE
    STARTADR quadword unsigned;  /*START ADDRESS
    MAJORID word unsigned;        /*MAJOR ID OF IMAGE HEADER
    constant MAJORID equals (%X3130) prefix IHX tag $K; /*^A/01/ MAJOR ID VALUE FOR CROSS LINKER
    MINORID word unsigned;        /*MINOR ID OF IMAGE HEADER
    constant MINORID equals (%X3130) prefix IHX tag $K; /*^A/01/ MINOR ID VALUE FOR CROSS LINKER
    constant MINORID1 equals (%X3130) prefix IHX tag $K; /*^A/01/ MINOR ID VALUE FOR CROSS LINKER WITH
    /*SYMBOL TABLE AND 3RD TRANSFER ADR
    IMGNAM character length 24;  /*IMAGE NAME
```

```
/*
/* THE FOLLOWING FIELDS ARE PRESENT FOR MINOR ID'S GREATER OR EQUAL TO "03"
/*
```

```
    DSTVBN longword unsigned;    /*DEBUG SYMBOL TABLE VBN
    GSTVBN longword unsigned;    /*GLOBAL SYMBOL TABLE VBN
    DSTBLKS word unsigned;       /*DEBUG SYMBOL TABLE BLOCKS
    GSTRECS word unsigned;       /*GLOBAL SYMBOL TABLE RECORD COUNT
    TFRADR3 longword unsigned;   /*THIRD TRANSFER ADDRESS
    constant 'LENGTH' equals . prefix IHX$ tag K; /*LENGTH OF CROSS LINKER HEADER
    constant 'LENGTH' equals . prefix IHX$ tag C; /*LENGTH OF CROSS LINKER HEADER
```

```
end IHXDEF;
```

```
end_module $IHXDEF;
```



```
module $IMPDEF;
```

```
/*+
/*      RMS32 IMPURE AREA OFFSET DEFINITIONS
/*
/*-
```

```
aggregate IMPDEF structure prefix IMP$;
```

```
  RMSSTATUS_OVERLAY union fill;
```

```
  RMSSTATUS word unsigned;
```

```
  RMSSTATUS BITS structure fill;
```

```
    IIOS bitfield;
```

```
    AST bitfield;
```

```
    TEMP1 bitfield;
```

```
    TEMP2 bitfield;
```

```
    IORUNDOWN bitfield;
```

```
    NOPOBUFS bitfield;
```

```
    RUH bitfield;
```

```
    RECOVERY bitfield;
```

```
    RUH_SYNCH bitfield;
```

```
  end RMSSTATUS_BITS;
```

```
/*
```

```
  constant ASYEFN equals 30 prefix IMP tag $C; /* EFN FOR ASYNC WAITS
```

```
  constant IOREFN equals 30 prefix IMP tag $C; /* EFN FOR IO RUNDOWN SYNCHRONIZATION
```

```
  constant ASYQIOEFN equals 31 prefix IMP tag $C; /* EFN FOR ASYNC QIOS
```

```
  constant SYNCEFN equals 27 prefix IMP tag $C; /* BASE EFN FOR SYNCHRONOUS QIO'S
```

```
/* (28, 29 ALSO USED)
```

```
  constant MBXEFN equals 26 prefix IMP tag $C; /* EFN FOR QIOS TO NETWORK MAILBOXES
```

```
/*
```

```
end RMSSTATUS_OVERLAY;
```

```
PROT byte unsigned;
```

```
FILL 1 byte fill prefix IMPDEF tag $$;
```

```
IOSEGADDR longword unsigned;
```

```
IOSEGLEN longword unsigned;
```

```
FREEPGLH longword unsigned dimension 2;
```

```
SAVED SP longword unsigned;
```

```
IFABTBL longword unsigned;
```

```
IRABTBL longword unsigned;
```

```
ENTERSEG word unsigned;
```

```
constant NPIOFILES equals 63 prefix IMP tag $C;
```

```
constant ENTERSEG equals 15 prefix IMP tag $C;
```

```
NUM IFABS word unsigned;
```

```
IFBTBLINK longword unsigned;
```

```
FILL 2 longword dimension 15 fill prefix IMPDEF tag $$;
```

```
IRBTBLINK longword unsigned;
```

```
end IMPDEF;
```

```
end_module $IMPDEF;
```

```
/* RMS OVERALL STATUS
```

```
/* SET IF THIS IS THE IMAGE
```

```
/* I.O SEGMENT
```

```
/* SET IF RUNNING AT EXEC AST LEVEL
```

```
/* TEMPORARY FLAG
```

```
/* "
```

```
/* SET IF IO RUNDOWN IN PROGRESS
```

```
/* SET IF RMS USE OF PO FOR IMAGE I/O DISABLED
```

```
/* Set if within RMS RU Handler
```

```
/* SET IF RECOVERY IN PROGRESS
```

```
/* SET IF RMS IO MUST SYNCH
```

```
/* WITH THE RU HANDLER
```

```
/* PROTECTION FOR I/O BUFFER PAGES
```

```
/* SPARE
```

```
/* ADDRESS OF FIRST FREE PAGE
```

```
/* IN THIS (IMAGE OR PROCESS)
```

```
/* I/O SEGMENT
```

```
/* ! OF FREE BYTES AT ABOVE ADDR
```

```
/* FREE PAGE LIST HEAD
```

```
/* SAVED VALUE OF SP AT ENTRY
```

```
/* IFAB TABLE ADDR
```

```
/* IRAB TABLE ADDR
```

```
/* ! OF SLOTS PER TABLE SEGMENT
```

```
/* ! OF PIO SEGMENT FILES
```

```
/* ! OF IIO SEGMENT SLOTS
```

```
/* PER INDEX TABLE SEGMENT
```

```
/* NUMBER OF IFABS & IRABS CURRENTLY ALLOCATED
```

```
/* START OF IFAB TABLE (LINK TO NEXT SEGMENT)
```

```
/* FIRST IFAB TABLE SEGMENT
```

```
/* START OF IRAB TABLE (LINK TO NEXT SEGMENT)
```



```
module $IOCDEF;
```

```
/**  
/*  
/* $IOCDEF - flag bits used in I/O database search routines.  
/*  
/*-
```

```
aggregate IOCDEF structure prefix IOC$;
```

PHY bitfield mask;	/* IOC\$V_PHY must be bit 0!!
TYPE bitfield mask;	/* physical device specified
CLASS bitfield mask;	/* device type name specified
LOCAL bitfield mask;	/* allocation class present
EXISTS bitfield mask;	/* search local devices only
"2P" bitfield mask;	/* device exists
ANY bitfield mask;	/* device is on UCB secondary path
MOUNT bitfield mask;	/* find any matching device
ALT bitfield mask;	/* find only mountable devices
NO TRANS bitfield mask;	/* alternate UCB found
ALLOC bitfield mask;	/* caller translated logical name
	/* allocate mountable device

```
end IOCDEF;
```

```
end_module $IOCDEF;
```



```
module $I0780DEF;
```

```
/*+
```

```
/* I/O SPACE LAYOUT FOR 11/780 CPU
```

```
/*-
```

```

constant IOBASE equals +%X20000000 prefix I0780$ tag AL; /*DEFINE CONSTANT ADDRESSES
constant PERNEX equals +%X2000 prefix I0780$ tag AL; /*START OF I/O SPACE
constant NNEX equals 16 prefix I0780$ tag AL; /*! BYTES OR REGISTER SPACE/NEXUS
constant UB0SP equals +%X20100000 prefix I0780$ tag AL; /*! OF NEXUSES
constant UB1SP equals +%X20140000 prefix I0780$ tag AL; /*ADDR OF UB 0 SPACE
constant UB2SP equals +%X20180000 prefix I0780$ tag AL; /*ADDR OF UB 1 SPACE
constant UB3SP equals +%X201C0000 prefix I0780$ tag AL; /*ADDR OF UB 2 SPACE
constant UB3SP equals +%X201C0000 prefix I0780$ tag AL; /*ADDR OF UB 3 SPACE

```

```
end_module $I0780DEF;
```

```
module $I0750DEF;
```

```
/*+
```

```
/* I/O SPACE LAYOUT FOR 11/750 CPU
```

```
/*-
```

```

constant IOBASE equals +%XF20000 prefix I0750$ tag AL; /*DEFINE CONSTANT ADDRESSES
constant PERNEX equals +%X2000 prefix I0750$ tag AL; /*START OF I/O SPACE FOR SLOT 16
constant UBBASE equals +%XF30000 prefix I0750$ tag AL; /*! BYTES OF REGISTER SPACE/NEXUS
constant MBBASE equals +%XF28000 prefix I0750$ tag AL; /*START OF UB 0 SPACE
constant FLOAT equals +%XF34000 prefix I0750$ tag AL; /*START OF MBO REGISTER SPACE
constant NNEX equals 16 prefix I0750$ tag AL; /*START OF FLOATING ADAPTER SPACE
constant UB0SP equals +%XFC0000 prefix I0750$ tag AL; /*! CONFIGURABLE NEXUSES
constant UB1SP equals +%XF80000 prefix I0750$ tag AL; /*ADDR OF UB0 SPACE
constant UB1SP equals +%XF80000 prefix I0750$ tag AL; /*ADDR OF UB1 SPACE
constant MEM0 equals 0 prefix I0750$C_ tag SL; /*FIXED ADAPTER ASSIGNMENTS FOR
constant MPM0 equals 1 prefix I0750$C_ tag SL; /* SLOTS 0-9:
constant MPM1 equals 2 prefix I0750$C_ tag SL; /* MEMORY CONTROLLER
constant MPM2 equals 3 prefix I0750$C_ tag SL; /* MULTIPOINT MEMORIES...
constant MBO equals 4 prefix I0750$C_ tag SL; /*
constant MB1 equals 5 prefix I0750$C_ tag SL; /* MASSBUS ADAPTERS...
constant MB2 equals 6 prefix I0750$C_ tag SL; /*
constant MB3 equals 7 prefix I0750$C_ tag SL; /*
constant UB0 equals 8 prefix I0750$C_ tag SL; /* UNIBUS 0
constant UB1 equals 9 prefix I0750$C_ tag SL; /* UNIBUS 1

```

```
end_module $I0750DEF;
```

```
module $I0730DEF;
```

```
/*+
```

```
/* I/O SPACE LAYOUT FOR 11/730 CPU
```

```
/*--
```

```

constant IOBASE equals +%XF20000 prefix I0730$ tag AL; /* START OF I/O SPACE
constant PERNEX equals +%X2000 prefix I0730$ tag AL; /* ! BYTES OF REGISTER SPACE/NEXUS
constant NNEX equals 16 prefix I0730$ tag AL; /* ! OF NEXUSES

```



```
constant UBOSP equals +XFC0000 prefix IO730$ tag AL; /* ADDR OF UB0 SPACE
```

```
end_module $IO730DEF;
```

```
module $IO790DEF;
```

```
/*+
/* I/O SPACE LAYOUT FOR 11/790 CPU
/*--
```

```
constant IOA0 equals +X20000000 prefix IO790$ tag AL; /*START OF I/O SPACE FOR ABUS ADAPTER 0
constant IOA1 equals +X22000000 prefix IO790$ tag AL; /*START OF I/O SPACE FOR ABUS ADAPTER 1
constant IOA2 equals +X24000000 prefix IO790$ tag AL; /*START OF I/O SPACE FOR ABUS ADAPTER 2
constant IOA3 equals +X26000000 prefix IO790$ tag AL; /*START OF I/O SPACE FOR ABUS ADAPTER 3
constant PERNEX equals +X2000 prefix IO790$ tag AL; /*! OF BYTES OF REGISTER SPACE/NEXUS
constant NNEX equals 16 prefix IO790$ tag AL; /*NUMBER OF NEXUS PER SBIA
constant UBOSP equals +X100000 prefix IO790$ tag AL; /* OFFSET OF UB 0 SPACE FROM BASE OF SBIA
constant IOACR equals +X80000 prefix IO790$ tag AL; /* OFFSET OF IO ADAPTER CR FROM BASE OF SBIA
constant PERABS equals +X2000000 prefix IO790$ tag AL; /* ADDRESS SPACE FOR EACH ABUS ADAPTER
```

```
constant SBIA equals 1 prefix IO790$ tag K; /* TYPE CODE FOR SBIA ADAPTER
constant SBIA equals 1 prefix IO790$ tag C; /* TYPE CODE FOR SBIA ADAPTER
```

```
end_module $IO790DEF;
```

```
module $IOUV1DEF;
```

```
/*+
/* I/O SPACE LAYOUT FOR MICRO-VAX I CPU
/*--
```

```
constant QBOSP equals +X20000000 prefix IOUV1$ tag AL; /* ADDR OF QBUS SPACE
```

```
end_module $IOUV1DEF;
```

```
module $IO8NNDEF;
```

```
/*+
/* I/O SPACE LAYOUT FOR 11/8NN CPU
/*--
```

```
/* NBIA Type code found in NAC field
```

```
constant NBIA equals +X10 prefix IO8NN$ tag K; /* NBIA type
constant NBIA equals +X10 prefix IO8NN$ tag C; /* NBIA type
```

```
/* Define constant addresses
```

```
constant IOBASE equals +X20000000 prefix IO8NN$ tag AL; /* Start of I/O space
/*
constant NBIA_0 equals +X20000000 prefix IO8NN$ tag AL; /* I/O space for first NBIA
constant NBIB_0 equals +X20000000 prefix IO8NN$ tag AL; /* I/O space 1st BI
constant NBIB_1 equals +X22000000 prefix IO8NN$ tag AL; /* I/O space 2nd BI
/*
constant NBIA_1 equals +X24000000 prefix IO8NN$ tag AL; /* I/O space for second NBIA
constant NBIB_2 equals +X24000000 prefix IO8NN$ tag AL; /* I/O space 3rd BI
```



```

constant NBIB_3 equals +XX26000000 prefix IO8NNS$ tag AL; /* I/O space 4th BI
/*
constant NMI_MEM equals +XX3E000000 prefix IO8NNS$ tag AL; /* Nautilus Memory
/*
constant NMI_CSRO equals +XX80000 prefix IO8NNS$ tag AL; /* offset to NMI CSRO
constant PERNMI equals +XX2000000 prefix IO8NNS$ tag AL; /* I/O space per NMI nexus
constant PERNBIA equals +XX40000 prefix IO8NNS$ tag AL; /* offset to next NBIA
constant PERNBIB equals +XX20000 prefix IO8NNS$ tag AL; /* offset to next NBIB
constant PERNEX equals +XX2000 prefix IO8NNS$ tag AL; /* # bytes of register space/nexus
constant NNEX equals 16 prefix IO8NNS$ tag AL; /* # of Nexuses per BI
constant UBOSP equals +XX100000 prefix IO8NNS$ tag AL; /* First unibus space

/* Define BI address space offsets and lengths

constant BRDCST equals +XX20000 prefix IO8NNS$ tag AL; /* offset to broadcast space
constant BTROM equals +XX40000 prefix IO8NNS$ tag AL; /* offset to boot rom space
constant NODESP equals +XX400000 prefix IO8NNS$ tag AL; /* offset to BI node window 0
constant NDSPER equals +XX40000 prefix IO8NNS$ tag AL; /* size of BI node window

end_module $IO8NNDEF;

module $IO8SSDEF;
/*+
/* I/O SPACE LAYOUT FOR 11/8SS CPU
/*-

constant IOBASE equals +XX20000000 prefix IO8SS$ tag AL; /* Define constant addresses
constant PERNEX equals +XX2000 prefix IO8SS$ tag AL; /* Base of I/O space
constant NNEX equals 16 prefix IO8SS$ tag AL; /* Size of Register Space/Node
constant BRDCST equals +XX20020000 prefix IO8SS$ tag AL; /* # of Nodes
constant BTROM equals +XX20040000 prefix IO8SS$ tag AL; /* Base of Broadcast Space
constant NDPRIV equals +XX20080000 prefix IO8SS$ tag AL; /* Base of Bootrom Space
constant PCNTL equals +XX20088000 prefix IO8SS$ tag AL; /* Base of Node Private Space
constant NIBUF equals +XX20090000 prefix IO8SS$ tag AL; /* Pcntl CSR in Node Private Space
constant EEPROM equals +XX20098000 prefix IO8SS$ tag AL; /* NI Packet Buffer in Node Private Space
constant NIDATA equals +XX200A0000 prefix IO8SS$ tag AL; /* EEPROM in Node Private Space
constant NIADDR equals +XX200A8000 prefix IO8SS$ tag AL; /* NI Data Register in Node Private Space
constant RCX50 equals +XX200B0000 prefix IO8SS$ tag AL; /* NI ADDR Register in Node Private Space
constant WATCH equals +XX200B8000 prefix IO8SS$ tag AL; /* RX50 Registers in Node Private Space
constant NODESP equals +XX20400000 prefix IO8SS$ tag AL; /* Watch Chip in Node Private Space
constant NDSPER equals +XX40000 prefix IO8SS$ tag AL; /* Node 0 Window Space
prefix IO8SS$ tag AL; /* Size of Window Space

end_module $IO8SSDEF;

```



```
module $IPLDEF;
```

```
/**
```

```
/* TEMPORARY PROCESSOR PRIORITY LEVEL DEFINITIONS
```

```
/*-
```

```
constant HWCLK equals 24 prefix IPL tag $;  
constant PERFMON equals 15 prefix IPL tag $;  
constant IOPOST equals 4 prefix IPL tag $;  
constant MAILBOX equals 11 prefix IPL tag $;  
constant POWER equals 31 prefix IPL tag $;  
constant QUEUEAST equals 6 prefix IPL tag $;  
constant SCHED equals 3 prefix IPL tag $;  
constant SYNCH equals 8 prefix IPL tag $;  
constant TIMER equals 8 prefix IPL tag $;  
constant TIMERFORK equals 7 prefix IPL tag $;  
constant ASTDEL equals 2 prefix IPL tag $;  
constant SCS equals 8 prefix IPL tag $;
```

```
end_module $IPLDEF;
```

```
/*
```

```
/*HARDWARE CLOCK LEVEL
```

```
/*PERFORMANCE MONITORING SYNCH LEVEL
```

```
/*I/O POST PROCESSING LEVEL
```

```
/*WRITE MAILBOX INTERLOCK LEVEL
```

```
/*POWERFAIL INTERLOCK LEVEL
```

```
/*QUEUE AST LEVEL
```

```
/*SCHEDULER LEVEL
```

```
/*SYSTEM DATA BASE SYNCHRONIZATION LEVEL
```

```
/*TIME QUEUE PROCESSING LEVEL
```

```
/*TIMER FORK INTERRUPT LEVEL
```

```
/*AST DELIVERY INTERRUPT
```

```
/*SCS SYNCHRONIZATION IPL
```



```
module $IRPDEF;
```

```
/**
/* IRP - I/O REQUEST PACKET
/*
/* I/O REQUEST PACKETS ARE CONSTRUCTED BY THE QUEUE I/O REQUEST SYSTEM
/* SERVICE. THE CONTENT OF AN I/O REQUEST PACKET DESCRIBES A FUNCTION TO
/* BE PERFORMED ON A DEVICE UNIT.
/*
/* NOTE: SEVERAL FIELDS OF THE IRP MUST BE AT THE SAME OFFSETS AS THEIR
/* CORRESPONDING FIELDS IN THE IRPE (SEE NEXT PAGE).
/*
/*-
```

```
aggregate IRPDEF structure prefix IRP$;
```

```
  IOQFL longword unsigned;
  IOQBL longword unsigned;
  SIZE word unsigned;
  TYPE byte unsigned;
  RMOD OVERLAY union fill;
    RMOD byte unsigned;
    RMOD BITS structure fill;
      MODE bitfield length 2;
    end RMOD BITS;
  end RMOD_OVERLAY;
  PID longword unsigned;
  AST longword unsigned;
  ASTPRM longword unsigned;
  WIND longword unsigned;
  UCB longword unsigned;
  FUNC OVERLAY union fill;
    FUNC word unsigned;
    FUNC BITS structure fill;
      FCODE bitfield mask length 6;
      FMOD bitfield length 10;
    end FUNC BITS;
  end FUNC_OVERLAY;
  EFN byte unsigned;
  PRI byte unsigned;
  IOSB longword unsigned;
  CHAN word unsigned;
  STS_OVERLAY union fill;
    STS word unsigned;
    STS BITS structure fill;
      BUFIO bitfield mask;
      FUNC bitfield mask;
      PAGIO bitfield mask;
      COMPLX bitfield mask;
      VIRTUAL bitfield mask;
      CHAINED bitfield mask;
      SWAPIO bitfield mask;
      DIAGBUF bitfield mask;
      PHYSIO bitfield mask;
      TERMIO bitfield mask;
      MBXIO bitfield mask;
```

```
/*I/O QUEUE FORWARD LINK
/*I/O QUEUE BACKWARD LINK
/*SIZE OF IRP IN BYTES
/*STRUCTURE TYPE FOR IRP

/*ACCESS MODE OF REQUEST
/* MODE SUBFIELD

/*PROCESS ID OF REQUESTING PROCESS
/*ADDRESS OF AST ROUTINE
/*AST PARAMETER
/*ADDRESS OF WINDOW BLOCK
/*ADDRESS OF DEVICE UCB

/*I/O FUNCTION CODE AND MODIFIERS
/* FUNCTION CODE FIELD
/* FUNCTION MODIFIER FIELD

/*EVENT FLAG NUMBER AND EVENT GROUP
/*BASE PRIORITY OF REQUESTING PROCESS
/*ADDRESS OF I/O STATUS DOUBLE LONGWORD
/*PROCESS I/O CHANNEL NUMBER

/*REQUEST STATUS

/* BUFFERED I/O FLAG ;THESE BITS
/* 1=>READ FUNCTION ;MUST BE ADJACENT
/* PAGING I/O FLAG ;AND IN ORDER
/* COMPLEX BUFFERED I/O
/* VIRTUAL I/O FUNCTION
/* CHAINED BUFFERED I/O OPERATION
/* SWAP I/O OPERATION
/* DIAGNOSTIC BUFFER ALLOCATED
/* PHYSICAL I/O
/* TERMINAL I/O (FOR SELECTING PRIORITY INC)
/* MAILBOX BUFFERED READ
```



```

    EXTEND bitfield mask;
    FILACP bitfield mask;
    MVIRP bitfield mask;
    JNL_REMREQ bitfield mask;
    KEY bitfield mask;
end STS BITS;
end STS OVERLAY;
SVAPTE longword unsigned;
BOFF word unsigned;
BCNT OVERLAY union fill;
    BCNT longword unsigned;
    BCNT word unsigned;
end BCNT OVERLAY;
FILL 1 word fill prefix IRPDEF tag $$;
IOST1 OVERLAY union fill;
    IOST1 longword unsigned;
    MEDIA longword unsigned;
end IOST1 OVERLAY;
IOST2 OVERLAY union fill;
    IOST2 longword unsigned;
    TT_TERM OVERLAY union fill;
        TT_TERM longword unsigned;
        CARCON byte unsigned;
    end TT_TERM OVERLAY;
end IOST2 OVERLAY;
NT_PRVMASK OVERLAY union fill;
    NT_PRVMASK quadword unsigned;
    STATION quadword unsigned;
    TT_STATE OVERLAY union fill;
        TT_STATE quadword unsigned;
        TT_STATE_FIELDS structure fill;
            ABCNT OVERLAY union fill;
                ABCNT longword unsigned;
                ABCNT word unsigned;
            end ABCNT OVERLAY;
            OBCNT OVERLAY union fill;
                OBCNT longword unsigned;
                OBCNT word unsigned;
            end OBCNT OVERLAY;
        end TT_STATE_FIELDS;
    end TT_STATE OVERLAY;
end NT_PRVMASK OVERLAY;
SEGVBN OVERLAY union fill;
    SEGVBN longword unsigned;
    JNL_SEQNO longword unsigned;
end SEGVBN OVERLAY;
DIAGBUF OVERLAY union fill;
    DIAGBUF longword unsigned;
    TT_PRMP word unsigned;
end DIAGBUF OVERLAY;
SEQNUM longword unsigned;
EXTEND longword unsigned;
ARB longword unsigned;
KEYDESC longword unsigned;

```

```

/* AN IRPE IS LINKED TO THIS IRP
/* FILE ACP I/O (BOTH DIOCNT AND BIOCNT)
/* MOUNT VERIFICATION IRP
/* REMOTE (SLAVE) REQUEST
/* KEY FOR ENCRYPTION

```

```

/*SYSTEM VIRTUAL ADDRESS OF FIRST PTE
/*BYTE OFFSET IN FIRST PAGE

```

```

/*BYTE COUNT OF TRANSFER
/* OLD WORD DEFINITION FOR COMPATIBILITY

```

```

/* ROUND UP TO NEXT LONGWORD

```

```

/*FIRST I/O STATUS LONGWORD (FOR I/O POST)
/*MEDIA ADDRESS

```

```

/*SECOND I/O STATUS LONGWORD

```

```

/*ADDRESS OF READ TERMINATORS MASK
/*CARRIAGE CONTROL

```

```

/* PRIVILEGE MASK FOR DECNET
/* STATION FIELD FOR DECNET DRIVERS

```

```

/* TERMINAL STATE DEFINITIONS

```

```

/* ACCUMULATED BYTES TRANSFERED
/* OLD WORD DEFINITION FOR COMPATIBILITY

```

```

/* ORIGINAL TRANSFER BYTE COUNT
/* OLD WORD DEFINITION FOR COMPATIBILITY

```

```

/*VIRTUAL BLOCK NUMBER OF CURRENT SEGMENT
/* SEQUENCE NUMBER IN JOURNAL

```

```

/* DIAGNOSTIC BUFFER ADDRESS
/* PROMPT SIZE

```

```

/* SEQUENCE NUMBER
/* ADDRESS OF IRPE
/* ACCESS RIGHTS BLOCK ADDRESS
/* ADDRESS OF ENCRYPTION DESCRIPTOR

```


/* Standard IRP must contain space for Class Driver CDRP fields.

constant CDRP equals .;
constant CDRP equals . tag C;

FQFL longword unsigned;
FQBL longword unsigned;
CDRPSIZE word unsigned;
CD TYPE byte unsigned;
FIPL byte unsigned;
FPC longword unsigned;
FR3 longword unsigned;
FR4 longword unsigned;
SAVD RTN longword unsigned;
MSG_BUF longword unsigned;
RSPID longword unsigned;
CDT longword unsigned;
RWCPTN longword unsigned;

/* Offset to the CDRP within the IRP
/* Offset to the CDRP within the IRP

/* Fork Queue FLINK
/* Fork Queue Blink
/* Size field for positive section only
/* Type, always of interest
/* Fork IPL
/* Fork PC
/* Fork R3
/* Fork R4
/* Saved return address from level 1 JSB
/* Address of allocated MSCP buffer
/* Allocated Request ID
/* Address of Connection Descriptor Table
/* RWAITCNT pointer

/* Extensions to the CDRP within the IRP

CDRP_EXTENSIONS union fill:

/* Block Transfer Extension

BLK_XFER_EXTENSION structure fill;
LBUFH_AD longword unsigned;
LBOFF longword unsigned;
RBUFH_AD longword unsigned;
RBOFF longword unsigned;
XCT_LEN longword unsigned;
constant BT_LEN equals .;
constant BT_LEN equals . tag C;
end BLK_XFER_EXTENSION;

/* Local BUFFER Handle Address
/* Local Byte Offset
/* Remote BUFFER Handle Address
/* Remote Byte Offset
/* Transfer length in bytes

/* Class Driver Extension

CLS_DRV_EXTENSION structure fill;
FILE_3 longword fill;
LBUFANDL character length 12;
UBARSRC longword unsigned;
DUTUFLAGS longword unsigned;
/* See CDRP definition for bit field definitions
DUTUCNTR word unsigned;
ENDMSGISZ word unsigned;
constant CD_LEN equals .;
constant CD_LEN equals . tag C;
end CLS_DRV_EXTENSION;

{ Skip local buffer handle address (above)
/* Local buffer handle
/* UNIBUS mapping resources allocated
/* Class driver status flags:

/* General purpose counter
/* Size of most recent MSCP end message

/* Connection management extension

CON_MGT_EXTENSION structure fill;
CNX_WORK_AREA union fill;
CNX_CLIENT_DATA structure fill;
VAL1 longword unsigned;

/* data value 1


```

        VAL2 longword unsigned;      /* data value 2
        VAL3 longword unsigned;      /* data value 3
        VAL4 longword unsigned;      /* data value 4
        VAL5 longword unsigned;      /* data value 5
        VAL6 longword unsigned;      /* data value 6
        VAL7 longword unsigned;      /* data value 7
        VAL8 longword unsigned;      /* data value 8
    end CNX_CLIENT_DATA;
    CNX_BLOCK_XFER structure fill;
        FILL_CBUFH_AD longword fill;  { filler for CDRPSL_LBUFH_AD
        FILL_VAL longword dimension 4; { filler for VAL2 through VAL5
        CNXSVAPTE longword unsigned;  /* Block SVAPTE
        CNXBOFF word unsigned;         /* Block buffer offset
        CNXBCNT longword unsigned;     /* Block xfer length
        CNXRMOD byte unsigned;         /* Block access mode
        CLTSTS byte unsigned;         /* A client's status field
    end CNX_BLOCK_XFER;
    end CNX_WORK_AREA;
    MSGBLD longword unsigned;          /* Address of MSG BUILD routine
    SAVEPC longword unsigned;          /* Caller's saved PC
    SENDSEQNM word unsigned;           /* Message sequence number
    CNXSTATE byte unsigned;            /* CNX message state
    constant (                          /* Possible states:
        NORMAL                          /* The standard case (particularity no block xfer)
        , REQUESTOR                     /* Block transfer requestor
        , PARTNER                       /* Block transfer partner
    ) equals 0 increment 1;
    FILL 5 byte fill;
    RETRSPID OVERLAY union fill;
        RETRSPID longword unsigned;    /* RSPID to return
        BTX longword unsigned;         /* BTX address
    end RETRSPID;
    VAL9 longword unsigned;            /* data value 9
    constant CM_LENGTH equals .;
    /*
    /* The following fields are only valid
    /* for long connection manager CDRPs.
    /*
    VAL10 longword unsigned;           /* data value 10
    constant CM_LONG_LENGTH equals.;
end CON_MGT_EXTENSION;

end CDRP_EXTENSIONS;

constant 'LENGTH' equals .;          /* LENGTH OF STANDARD IRP
constant 'LENGTH' equals . tag C;    /* LENGTH OF STANDARD IRP

end IRPDEF;

end_module $IRPDEF;

```



```
module $IRPEDEF;
```

```
/**
/* IRPE - I/O REQUEST PACKET EXTENSION
/*
/* I/O REQUEST PACKET EXTENSIONS ARE USED TO HOLD ADDITIONAL INFORMATION
/* ABOUT I/O REQUESTS FOR DEVICES THAT REQUIRE MORE CONTEXT THAN CAN FIT INTO
/* THE STANDARD IRP. IRPE'S ARE BUILT AND LINKED ONTO IRP'S BY DEVICE
/* DRIVER FDT ROUTINES. ANY FIELDS THAT ARE NOT DEFINED IN THIS STRUCTURE
/* MAY BE USED TO HOLD DRIVER DEPENDENT DATA.
/*
/* THE CURRENTLY DEFINED FIELDS IN THE IRPE WERE POSITIONED SO THAT THE
/* PACKET COULD BE USED AS A FORK BLOCK. THIS SHOULD BE KEPT IN MIND IF
/* AND WHEN NEW FIELDS ARE DEFINED.
/*
/* THE FIELDS DEFINED HERE MUST BE AT THE SAME OFFSETS AS THEIR CORRESPONDING
/* FIELDS IN THE IRP (SEE PREVIOUS PAGE).
/*
/*--
```

```
aggregate IRPEDEF structure prefix IRPES;
```

```
  FILL_1 longword fill prefix IRPEDEF tag $$; /* SPARE LONGWORD
  FILL_2 longword fill prefix IRPEDEF tag $$; /* SPARE LONGWORD
  SIZE word unsigned; /* SIZE OF IRPE IN BYTES
  TYPE byte unsigned; /* STRUCTURE TYPE FOR IRPE
  FILL_3 byte fill prefix IRPEDEF tag $$; /* SPARE BYTE
  FILL_4 longword dimension 7 fill prefix IRPEDEF tag $$; /* 7 SPARE LONGWORDS
  FILL_5 word fill prefix IRPEDEF tag $$; /* SPARE WORD
  STS_OVERLAY union fill;
    STS word unsigned; /* STATUS
    STS_BITS structure fill;
      FILL_6 bitfield length 11 fill prefix IRPEDEF tag $$; /* SKIP OVER 11 BITS
      EXTEND bitfield mask; /* ANOTHER IRPE IS LINKED TO THIS ONE
    end STS_BITS;
  end STS_OVERLAY;
  SVAPTE1 longword unsigned; /* SYSTEM VIR. ADDR. OF PTE FOR REGION 1
  BOFF1 word unsigned; /* BYTE OFFSET FOR REGION 1
  FILL_7 word fill prefix IRPEDEF tag $$; /* SPARE WORD
  BCNT1 longword unsigned; /* BYTE COUNT FOR REGION 1
  SVAPTE2 longword unsigned; /* SYSTEM VIR. ADDR. OF PTE FOR REGION 2
  BOFF2 word unsigned; /* BYTE OFFSET FOR REGION 2
  FILL_8 word fill prefix IRPEDEF tag $$; /* SPARE WORD
  BCNT2 longword unsigned; /* BYTE COUNT FOR REGION 2
  FILL_9 longword dimension 4 fill prefix IRPEDEF tag $$; /* 4 SPARE LONGWORDS
  EXTEND longword unsigned; /* ADDRESS OF NEXT IRPE
  constant "LENGTH" equals . prefix IRPES tag K; /* LENGTH OF IRPE
  constant "LENGTH" equals . prefix IRPES tag C; /* LENGTH OF IRPE
end IRPEDEF;
```

```
end_module $IRPEDEF;
```



```
module $ISDDEF;
```

```
/**
```

```
/* IMAGE SECTION DESCRIPTOR DEFINITIONS
```

```
/*-
```

```
aggregate ISDDEF structure prefix ISD$;
```

```
SIZE word unsigned;
```

```
/*SIZE IN BYTES OF THIS ISD
```

```
PAGCNT word unsigned;
```

```
/*! OF PAGES DESCRIBED BY THIS ISD
```

```
VPNPFC OVERLAY union fill;
```

```
/*VPN & PFC VIELDS
```

```
VPNPFC longword unsigned;
```

```
VPNPFC BITS0 structure fill;
```

```
VPN bitfield length 21;
```

```
/* STARTING VIRTUAL PAGE NUMBER
```

```
P1 bitfield;
```

```
/* P1 SPACE
```

```
SYSTEM bitfield;
```

```
/* SYSTEM SPACE
```

```
FILL 1 bitfield fill prefix ISDDEF tag $$;
```

```
/* SPARE
```

```
PFC bitfield length 8;
```

```
/* PAGE FAULT CLUSTER
```

```
end VNPFC BITS0;
```

```
VPNPFC BITS1 structure fill;
```

```
/* VIRTUAL PAGE INCLUDING P1 & S
```

```
VPG bitfield length 23;
```

```
end VNPFC BITS1;
```

```
VPNPFC FIELDS2 structure fill;
```

```
FILL 4 byte dimension 3 fill prefix ISDDEF tag $$;
```

```
/*PAGE FAULT CLUSTER
```

```
PFC byte unsigned;
```

```
end VNPFC FIELDS2;
```

```
end VNPFC OVERLAY;
```

```
FLAGS OVERLAY union fill;
```

```
FLAGS longword unsigned;
```

```
/*FLAGS AND ISD TYPE
```

```
constant LENDZRO equals . prefix ISD$ tag K;
```

```
/*LENGTH OF DEMAND ZERO ISD
```

```
constant LENDZRO equals . prefix ISD$ tag C;
```

```
/*LENGTH OF DEMAND ZERO ISD
```

```
FLAGS BITS structure fill;
```

```
GBL bitfield mask;
```

```
/* GLOBAL
```

```
CRF bitfield mask;
```

```
/* COPY ON REFERENCE
```

```
DZRO bitfield mask;
```

```
/* DEMAND ZERO PAGE
```

```
WRT bitfield mask;
```

```
/* WRITABLE
```

```
MATCHCTL bitfield mask length 3;
```

```
/* IDENT MATCH CONTROL FIELD
```

```
LASTCLU bitfield mask;
```

```
/* ISD IS PART OF LAST PO SPACE CLUSTER
```

```
COPYALWAY bitfield mask;
```

```
/* COPY ALWAYS FROM USER IMAGE
```

```
"BASED" bitfield mask;
```

```
/* ISECT IS BASED
```

```
FIXUPVEC bitfield mask;
```

```
/* ISECT IS FIXUP SECTION
```

```
FILL 2 bitfield length 6 fill prefix ISDDEF tag $$; /* UNUSED, RESERVED FOR FUTURE USE
```

```
/* VECTOR CONTAINED IN IMAGE SECTION
```

```
VECTOR bitfield mask;
```

```
/* IMAGE SECTION IS PROTECTED
```

```
PROTECT bitfield mask;
```

```
FILL 3 bitfield length 5 fill prefix ISDDEF tag $$; /* UNUSED, RESERVED FOR FUTURE USE
```

```
end FLAGS BITS;
```

```
constant FLAGSI2 equals 24 prefix ISD tag $$; /* NUMBER OF FLAG BITS, ISD TYPE EXCLUDED
```

```
FLAGS FIELDS structure fill;
```

```
FILL 5 byte dimension 3 fill prefix ISDDEF tag $$;
```

```
/*ISD TYPE CODE
```

```
TYPE byte unsigned;
```

```
end FLAGS FIELDS;
```

```
end FLAGS OVERLAY;
```

```
VBN longword unsigned;
```

```
/*BASE VIRTUAL BLOCK NUMBER
```

```
constant LENPRIV equals . prefix ISD$ tag K;
```

```
/*LENGTH OF PRIVATE ISD
```

```
constant LENPRIV equals . prefix ISD$ tag C;
```

```
/*LENGTH OF PRIVATE ISD
```

```
"IDENT" longword unsigned;
```

```
/*IDENT FOR GLOBAL SECTION
```



```
GBLNAM character length 44;
constant LENGLBL equals .-28 prefix ISD$ tag K; /*GLOBAL NAME COUNTED STRING
constant LENGLBL equals .-28 prefix ISD$ tag C; /*LENGTH OF OLD GLOBAL ISD
constant MAXLENGLBL equals . prefix ISD$ tag K; /*LENGTH OF OLD GLOBAL ISD
constant MAXLENGLBL equals . prefix ISD$ tag C; /*MAX LENGTH OF NEW GLOBAL ISD
/*MAX LENGTH OF NEW GLOBAL ISD

/*+
/* MATCH CONTROL VIELD VALUES
/*-

constant(
    MATALL
    , MATEQU
    , MATLEQ
    , MATNEV
    ) equals 0 increment 1 prefix ISD tag $K; /*BASE OF ZERO , INCR 1
/*MATCH ALWAYS, USE GLOBAL SECTION
/*MATCH IF ISD$L_IDENT EQU GBL ID
/*MATCH IF ISD$L_IDENT LEQ GBL ID
/*MATCH NEVER, USE PRIVATE COPY

/*+
/* ISD TYPE FIELD DEFINITIONS
/*-

constant NORMAL equals 0 prefix ISD tag $K; /*NORMAL PROGRAM IMAGE SECTION
constant SHRFXD equals 1 prefix ISD tag $K; /*NO SPECIAL ACTION REQUIRED
constant PRVFXD equals 2 prefix ISD tag $K; /*SHAREABLE FIXED SECTION
constant SHRPIC equals 3 prefix ISD tag $K; /*PRIVATE FIXED SECTION
constant PRVPIC equals 4 prefix ISD tag $K; /*SHAREABLE PIC SECTION
constant USRSTACK equals (256-3) prefix ISD tag $K; /*PRIVATE PIC SECTION
/*USER STACK SECTION

end ISDDEF;
end_module $ISDDEF;
```



```

module ISDOLDDEF;
/*+
/* OLD IMAGE SECTION DESCRIPTOR DEFINITIONS
/*-

aggregate ISDOLDDEF structure prefix ISD_:
    SIZE word unsigned; /*SIZE IN BYTES OF THIS ISD
    PAGCNT word unsigned; /*! OF PAGES DESCRIBED BY THIS ISD
    VPNPFC OVERLAY union;
        VPNPFC longword unsigned; /*VPN & PFC VIELDS
        VPNPFC BITS0 structure;
            VPN bitfield length 21; /* STARTING VIRTUAL PAGE NUMBER
            P1 bitfield; /* P1 SPACE
            SYSTEM bitfield; /* SYSTEM SPACE
            FILL 1 bitfield fill prefix ISDOLDDEF tag --; /* SPARE
            PFC bitfield length 8; /* PAGE FAULT CLUSTER
        end VPNPFC BITS0;
        VPNPFC BITS1 structure;
            VPG bitfield length 23; /* VIRTUAL PAGE INCLUDING P1 & S
        end VPNPFC BITS1;
        VPNPFC FIELDS2 structure;
            FILL 4 byte dimension 3 fill prefix ISDOLDDEF tag ;
            PFC byte unsigned; /*PAGE FAULT CLUSTER
        end VPNPFC FIELDS2;
    end VPNPFC OVERLAY;
    FLAGS OVERLAY union;
        FLAGS longword unsigned; /*FLAGS AND ISD TYPE
        constant LENDZRO equals . prefix ISD_ tag K; /*LENGTH OF DEMAND ZERO ISD
        constant LENDZRO equals . prefix ISD_ tag C; /*LENGTH OF DEMAND ZERO ISD
        FLAGS BITS structure;
            GBL bitfield mask; /* GLOBAL
            CRF bitfield mask; /* COPY ON REFERENCE
            DZRO bitfield mask; /* DEMAND ZERO PAGE
            WRT bitfield mask; /* WRITABLE
            MATCHCTL bitfield mask length 3; /* IDENT MATCH CONTROL FIELD
            LASTCLU bitfield mask; /* ISD IS PART OF LAST PO SPACE CLUSTER
            COPYALWAY bitfield mask; /* COPY ALWAYS FROM USER IMAGE
            "BASED" bitfield mask; /* ISECT IS BASED
            FIXUPVEC bitfield mask; /* ISECT IS FIXUP SECTION
            FILL 2 bitfield length 6 fill prefix ISDOLDDEF tag ; /* UNUSED, RESERVED FOR FUTURE USE
            VECTOR bitfield mask; /* VECTOR CONTAINED IN IMAGE SECTION
            PROTECT bitfield mask; /* IMAGE SECTION IS PROTECTED
            FILL 3 bitfield length 5 fill prefix ISDOLDDEF tag __; /* UNUSED, RESERVED FOR FUTURE USE
        end FLAGS BITS;
        constant FLAGSIZ equals 24 prefix ISD tag _S; /* NUMBER OF FLAG BITS, ISD TYPE EXCLUDED
        FLAGS FIELDS structure;
            FILL 5 byte dimension 3 fill prefix ISDOLDDEF tag ;
            TYPE byte unsigned; /*ISD TYPE CODE
        end FLAGS FIELDS;
    end FLAGS OVERLAY;
    VBN longword unsigned; /*BASE VIRTUAL BLOCK NUMBER
    constant LENPRIV equals . prefix ISD_ tag K; /*LENGTH OF PRIVATE ISD
    constant LENPRIV equals . prefix ISD_ tag C; /*LENGTH OF PRIVATE ISD
    "IDENT" longword unsigned; /*IDENT FOR GLOBAL SECTION
    GBLNAM character length 16; /*GLOBAL NAME COUNTED STRING
    constant LENGLBL equals . prefix ISD_ tag K; /*LENGTH OF GLOBAL ISD

```



```
constant LENGLBL equals . prefix ISD_ tag C;      /*LENGTH OF GLOBAL ISD

/*+
/* MATCH CONTROL VIELD VALUES
/*-

constant(
    MATALL
    , MATEQU
    , MATLEQ
    , MATNEV
    ) equals 0 increment 1 prefix ISD tag _K;

/*+
/* ISD TYPE FIELD DEFINITIONS
/*-

constant NORMAL equals 0 prefix ISD tag _K;      /*NORMAL PROGRAM IMAGE SECTION
/*NO SPECIAL ACTION REQUIRED
constant SHRFXD equals 1 prefix ISD tag _K;      /*SHAREABLE FIXED SECTION
constant PRVFXD equals 2 prefix ISD tag _K;      /*PRIVATE FIXED SECTION
constant SHRPIC equals 3 prefix ISD tag _K;      /*SHAREABLE PIC SECTION
constant PRVPIC equals 4 prefix ISD tag _K;      /*PRIVATE PIC SECTION
constant USRSTACK equals (256-3) prefix ISD tag _K; /*USER STACK SECTION

end ISDOLDDEF;
end_module ISDOLDDEF;
```



```
module $JIBDEF;
```

```
/*+
/* Job Information Block - Structure containing common context for a set
/* of related processes.
/*
/* Note: The Executive module SYSCREPRC assumes that the job mount list head
/* precedes the username field in the JIB.
/*
/*-
```

```
aggregate JIBDEF structure prefix JIB$;
```

```

MTLFL longword unsigned; /* Job mount list head forward link
MTLBL longword unsigned; /* Job mount list head back link
SIZE word unsigned; /* Size of structure in bytes
TYPE byte unsigned; /* Structure type code
DAYTYPES byte unsigned; /* Set bits 0-6 flag non-prime days of week
USERNAME character length 12; /* User name for easy access
ACCOUNT character length 8; /* Account name for resident access
BYTCNT longword unsigned; /* Buffered I/O byte count avail
BYTLM longword unsigned; /* Original value for Byte count
PBYTCNT longword unsigned; /* Paged pool byte count remaining
PBYTLM longword unsigned; /* Paged pool byte limit
FILCNT word unsigned; /* Open File count remaining
FILLM word unsigned; /* Open file limit
TQCNT word unsigned; /* Timer queue entry count remaining
TQLM word unsigned; /* Timer queue entry limit
PGFLQUOTA longword unsigned; /* Paging file quota
PGFLCNT longword unsigned; /* Paging file limit
CPULIM longword unsigned; /* CPU time quota remaining
PRCNT word unsigned; /* Count of subprocesses existing
PRCLIM word unsigned; /* Limit on number of subprocesses
SHRFCNT word unsigned; /* Shared file block count remainig
SHRFLIM word unsigned; /* Shared file count limit
ENQCNT word unsigned; /* Enqueue count avail
ENQLM word unsigned; /* Enqueue limit
MAXJOBS word unsigned; /* Max jobs limit on user
MAXDETACH word unsigned; /* Max detached processes for user
MPID longword unsigned; /* PID of master process
JLNAMFL longword unsigned; /* Forward link for job-wide logical names
JLNAMBL longword unsigned; /* Back link for job-wide logical names
PDAYHOURS longword unsigned; /* Field describing primary day access
ODAYHOURS longword unsigned; /* Field describing off day access
JOBTYP byte unsigned; /* Job origin type
    constant (
        DETACHED
        , NETWORK
        , BATCH
        , LOCAL
        , DIALUP
        , REMOTE
    ) equals 0 increment 1 tag C;
FILL 4 byte dimension 3 fill tag $$;
ORG_BYTLM longword unsigned; /* Original BYTLM
ORG_PBYTLM longword unsigned; /* Original PBYTLM
constant "LENGTH" equals . prefix JIB$ tag K; /* Structure length
constant "LENGTH" equals . prefix JIB$ tag C; /* Structure length
```


SYSDEFFL.SDL;1

16-SEP-1984 16:45:23.65¹ Page 49

end JIBDEF;

end_module \$JIBDEF;

SYS

mod

/**

/*

/*

/*

/*

/*-

agg


```
module $KDZDEF;
```

```
/*
/* KDZ11 Offset Definitions for Registers Accessible Through BI Node Private
/* Space. Note that in making these registers available in virtual space,
/* we have only mapped real registers. Therefore these virtual offsets are
/* different than the hardware physical offsets.
/*-
```

```
aggregate KDZDEF structure prefix KDZ$;
```

```
/*
/* BIIC registers - here we reserve space for the 256 bytes that these
/* registers occupy and we also fill out the virtual page to
/* 512 bytes so that other items appear on page boundaries.
/* Being able to address the BIIC via these virtual addresses
/* allows a Scorpio CPU to determine its own node number.
/* That is, a reference here is via node private space and
/* always addresses a nodes own registers via a loop back
/* request.
```

```
BIICBASE byte unsigned; /*BIIC register Base
FILL_1 byte dimension 511 fill prefix KDZDEF tag $$; /* Fill out to page.
```

```
/*
/* Port Control CSR register
/*
```

```
PCNTL_OVERLAY union fill;
```

```
PCNTL_longword unsigned;
```

```
PCNTL_BITS structure fill;
```

```
PCNTL_RSTRT bitfield mask;
```

```
PCNTL_PHYLOG bitfield mask;
```

```
PCNTL_SECENB bitfield mask;
```

```
PCNTL_STINIT bitfield mask;
```

```
PCNTL_STFAST bitfield mask;
```

```
PCNTL_ENBAPT bitfield mask;
```

```
PCNTL_STPASS bitfield mask;
```

```
PCNTL_RUN bitfield mask;
```

```
FILL_2 bitfield fill prefix KDZ tag $$;
```

```
PCNTL_CLREVL bitfield mask;
```

```
PCNTL_WRMEM bitfield mask;
```

```
PCNTL_EV4 bitfield mask;
```

```
PCNTL_EV3 bitfield mask;
```

```
PCNTL_EV2 bitfield mask;
```

```
PCNTL_EV1 bitfield mask;
```

```
PCNTL_EV0 bitfield mask;
```

```
PCNTL_WWPO bitfield mask;
```

```
FILL_3 bitfield length 2 fill prefix KDZ tag $$; /*
```

```
PCNTL_RXDIS bitfield mask;
```

```
PCNTL_NIDIS bitfield mask;
```

```
PCNTL_CNSLIE bitfield mask;
```

```
/*Port Control CSR Register
```

```
/* Port Controller CSR
```

```
/* (RO) Front Panel Switch
```

```
/* selecting RSTRT/HALT
```

```
/* (RO) Backplane Bit
```

```
/* selecting PHYS/LOG Console
```

```
/* (RO) Front Panel Switch
```

```
/* to lock out console input
```

```
/* Self-Test INIT.
```

```
/* (RO) Backplane bit to
```

```
/* select Fast Self-Test.
```

```
/* Enable APT.
```

```
/* Self-Test Pass.
```

```
/* 1=>Program mode,0=>Console
```

```
/*
```

```
/* Clear Event Lock
```

```
/* Write Memory Bit
```

```
/* Event Bits - These
```

```
/* RO bits are event
```

```
/* codes from BIIC to
```

```
/* allow CPU to monitor
```

```
/* BI status
```

```
/* Write Wrong Parity Odd
```

```
/* Disable RX50
```

```
/* Disable NI Lance
```

```
/* Console Interrupt Enable
```

end

end


```
PCNTL_CNSLCL bitfield mask; /* Clear Console Interrupt
PCNTL_CNSLIN bitfield mask; /* Console Interrupt RCVD
PCNTL_WWPE bitfield mask; /* Write Wrong Parity Even
PCNTL_RXDONE bitfield mask; /* RX Done Interrupt
PCNTL_RXSTAT bitfield mask; /* RX Status Interrupt
PCNTL_CLRIPI bitfield mask; /* Clear IP Interrupt
PCNTL_IPINTR bitfield mask; /* IP Interrupt RCVD
PCNTL_CRDIE bitfield mask; /* CRD Interrupt Enable
PCNTL_CLRCRD bitfield mask; /* Clear CRD Interrupt
PCNTL_CRDINT bitfield mask; /* CRD Interrupt RCVD
end PCNTL_BITS;

end PCNTL_OVERLAY;

FILL_4 byte dimension 508 fill prefix KDZDEF tag $$; /* Fill out page

/*
/* NI Packet Buffer
/*
    NIBUF byte unsigned; /*NI Packet Buffer Base
    FILL_5 byte dimension 32767 fill prefix KDZDEF tag $$; /* Fill out to 32KB

/*
/* EEPROM
/*
    EEPROM byte unsigned; /*EEPROM Base
    FILL_6 byte dimension 8191 fill prefix KDZDEF tag $$; /* Fill out to 8KB

/*
/* NI Data Register
/*
    NIDATA longword unsigned; /* NI Data Register
    FILL_7 byte dimension 508 fill prefix KDZDEF tag $$; /* Fill out page

/*
/* NI Address Register
/*
    NIADDR longword unsigned; /* NI Address Register
    FILL_8 byte dimension 508 fill prefix KDZDEF tag $$; /* Fill out page

/*
/* RCX50 Registers
/*
    RCX50 byte unsigned; /* RCX50 Registers
    FILL_9 byte dimension 511 fill prefix KDZDEF tag $$; /* Fill out page

/*
/* Watch Chip Registers
/*
    WATCH byte unsigned; /* Watch Chip Registers
```

```
mod
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
```

agg

```
/*
/*
/*
```

```
/*
/*
/*
```

```
end
end
```


SYSDEFFL.SDL;1

16-SEP-1984 16:45:23.65^{M 1} Page 52

FILL_10 byte dimension 511 fill prefix KDZDEF tag \$\$; /* Fill out page
end KDZDEF;
end_module \$KDZDEF;

SYS

mod

{+ LI

{ T

{ T

{ E

{-

agg

end

{+ LI

{ T

{ T

{-

agg

con

end


```
module $KFDDEF;
```

```
/*  
/* KNOWN FILE DEVICE AND DIRECTORY BLOCK DEFINITIONS  
/*
```

```
aggregate KFDDEF structure prefix KFDS;
```

LINK longword unsigned;	/* Device, Directory, Extension (KFD) list link
KFELIST longword unsigned;	/* Ordered Known file entry list header
SIZE word unsigned;	/* Size of block
TYPE byte unsigned;	/* Structure type
SPARE byte unsigned;	/* spare
REFCNT word unsigned;	/* Number of KFE's with same KFD
DEVLEN byte unsigned;	/* Length of Device string
DIRLEN byte unsigned;	/* Length of Directory string
DDTSTRLEN byte unsigned;	/* Length of Device, Directory, Type (DDT) string
constant 'LENGTH' equals . prefix KFDS tag C;	/* Length of fixed area of kfd entry
DDTSTR character length 0;	/* Offset to DDT string

```
end KFDDEF;
```

```
end_module $KFDDEF;
```

```
{+  
{ LI  
{ T  
{ T  
{-  
agg
```

```
end  
{+  
{ LI  
{ T  
{ A  
{-  
agg
```



```
module $KFEDEF;
```

```
/*  
/* KNOWN FILE ENTRY DEFINITIONS  
/*
```

```
aggregate KFEDEF structure prefix KFES;
```

```
  HSHLNK longword unsigned;  
  KFELINK longword unsigned;  
  SIZE word unsigned;  
  TYPE byte unsigned;  
  HSHIDX byte unsigned;  
  KFD longword unsigned;  
  FLAGS OVERLAY union;  
    FLAGS word unsigned;  
    FLAGS BITS structure;  
      PROTECT bitfield mask;  
      LIM bitfield mask;  
      PROCPRIV bitfield mask;  
      OPEN bitfield mask;  
      HDRRES bitfield mask;  
      SHARED bitfield mask;  
      SHMIDENT bitfield mask;  
      COMPATMOD bitfield mask;  
      NOPURGE bitfield mask;  
      ACCOUNT bitfield mask;  
      WRITEABLE bitfield mask;  
      EXEONLY bitfield mask;  
    end FLAGS BITS;  
  end FLAGS_OVERLAY;  
  GBLSECCNT word unsigned;  
  USECNT longword unsigned;  
  WINDOW OVERLAY union;  
    WCB longword unsigned;  
    WINDOW FIELDS structure;  
      FID_OVERLAY union;  
        FID word unsigned;  
        FID_NUM word unsigned;  
      end FID_OVERLAY;  
      FID_SEQ word unsigned;  
    end WINDOW FIELDS;  
  end WINDOW_OVERLAY;  
  IMGHDR OVERLAY union;  
    IMGHDR longword unsigned;  
    FID RVN word unsigned;  
  end IMGHDR_OVERLAY;  
  PROCPRIV quadword unsigned;  
  MATCHCTL byte unsigned;  
  FILL 4 byte fill prefix KFEDEF tag $$;  
  AMECOD word unsigned;  
  "IDENT" longword unsigned;  
  ORB longword unsigned;  
  SHRCNT word unsigned;  
  FILNAMLEN byte unsigned;  
  constant "LENGTH" equals . prefix KFES tag K;  
  constant "LENGTH" equals . prefix KFES tag C;
```

```
/* Known file Hash table link  
/* Ordered Known file entry list link  
/* Size of block  
/* Structure type  
/* KFE hash table index (index into vector of HSHQ's)  
/* Device, Directory, Type block
```

```
/* Flags word  
/* Known file was installed protected  
/* Linkable image  
/* Use process privilege mask  
/* Image installed /OPEN  
/* Image header block is resident  
/* Image is shared  
/* Shared memory ident already set  
/* Image is compatability mode  
/* Image entry may not be purged  
/* Image level accounting  
/* Global sections are writeable  
/* Image has only execute access allowed
```

```
/* Global section count if shared  
/* Usage counter
```

```
/* WCB address if open
```

```
/* File id  
/* File number field of file id  
/* File sequence number field of file id
```

```
/* Image header address if resident  
/* Relative volume number field of file id
```

```
/* Process privilege mask  
/* Global section match control  
/* spare byte  
/* Image header code specifying AME  
/* Global section ident value  
/* Address of Object Rights Block  
/* High water mark for sharing  
/* Length of file name  
/* Length of fixed area of KFE entry  
/* Length of fixed area of KFE entry
```



```
constant MAXLEN equals .+39 prefix KFES tag K; /* Max KFE length (includes max filename)
constant MAXLEN equals .+39 prefix KFES tag C; /* Max KFE length (includes max filename)
FILNAM character length 0; /* Offset to file name
```

```
end KFEDEF;
```

```
end_module $KFEDEF;
```



```
module $KFHDEF;
```

```
/*  
/* KNOWN FILE IMAGE HEADER DEFINITIONS *** obsolete, to be removed ***  
/*
```

```
aggregate KFHDEF structure prefix KFHS;
```

```
    BUFEND longword unsigned;
```

```
    KFIADR longword unsigned;
```

```
    SIZE word unsigned;
```

```
    TYPE byte unsigned;
```

```
    FILL_1 byte fill prefix KFHDEF tag $$;
```

```
    constant "LENGTH" equals . prefix KFHS tag K;
```

```
    constant "LENGTH" equals . prefix KFHS tag C;
```

```
/*ADDRESS OF END OF KNOWN FILE HEADER
```

```
/*ADDRESS OF ASSOCIATED KNOWN FILE ENTRY
```

```
/*SIZE OF DYNAMIC STRUCTURE
```

```
/*DYNAMIC STRUCTURE TYPE
```

```
/*SPARE BYTE
```

```
/*LENGTH OF OVERHEAD AREA
```

```
/*LENGTH OF OVERHEAD AREA
```

```
/*  
/* THE REMAINDER OF THIS STRUCTURE CONTAINS THE IMAGE HEADER OF THE  
/* SPECIFIED KNOWN FILE. THE LOCATION KFISL_IMGHDR IN THE KNOWN FILE  
/* ENTRY POINTS KFHS_C_LENGTH INTO THIS STRUCTURE, I.E AT THE IMAGE HEADER  
/* ITSELF.  
/*
```

```
end KFHDEF;
```

```
end_module $KFHDEF;
```



```
module $KFIDEF;
```

```
/*
```

```
/* KNOWN FILE ENTRY DEFINITIONS
```

```
/*
```

```
*** obsolete, to be removed ***
```

```
aggregate KFIDEF structure prefix KFIS;
```

```
  KFIQFL longword unsigned;
```

```
  KFIQBL longword unsigned;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  KFICTL OVERLAY union fill;
```

```
    KFICTL byte unsigned;
```

```
    KFICTL BITS structure fill;
```

```
      KFIHD bitfield mask;
```

```
      FILIDOPEN bitfield mask;
```

```
      DONOTOPEN bitfield mask;
```

```
      FILL 1 bitfield length 3 fill prefix KFIDEF
```

```
      NOREPLACE bitfield mask;
```

```
      MARKDEL bitfield mask;
```

```
    end KFICTL BITS;
```

```
  end KFICTL OVERLAY;
```

```
  DEVUCB OVERLAY union fill;
```

```
    DEVUCB byte unsigned;
```

```
    DEVNAM byte unsigned;
```

```
  end DEVUCB OVERLAY;
```

```
  DIRNAM byte unsigned;
```

```
  FILNAM byte unsigned;
```

```
  TYPNAM byte unsigned;
```

```
  REFCNT word unsigned;
```

```
  KFIQNUM byte unsigned;
```

```
  KFISEQ OVERLAY union fill;
```

```
    KFISEQ byte unsigned;
```

```
    constant KFIHDLEN equals . prefix KFIS tag K;
```

```
    constant KFIHDLEN equals . prefix KFIS tag C;
```

```
    KFISEQ BITS structure fill;
```

```
      KFISEQ bitfield mask length 2;
```

```
    end KFISEQ BITS;
```

```
  end KFISEQ OVERLAY;
```

```
  FLAGS OVERLAY union fill;
```

```
    FLAGS word unsigned;
```

```
    FLAGS BITS structure fill;
```

```
      KP_OPEN bitfield mask;
```

```
      KP_RESHDR bitfield mask;
```

```
      KP_SHARED bitfield mask;
```

```
      PROTECT bitfield mask;
```

```
      FILL 2 bitfield length 2 fill prefix KFIDEF
```

```
      LIM bitfield mask;
```

```
      PROCPRIV bitfield mask;
```

```
      IS_RESHDR bitfield mask;
```

```
      IS_SHARED bitfield mask;
```

```
      FILL 3 bitfield length 4 fill prefix KFIDEF
```

```
      SHMIDENT bitfield mask;
```

```
      COMPATMOD bitfield mask;
```

```
    end FLAGS BITS;
```

```
  end FLAGS OVERLAY;
```

```
/*KNOWN FILE QUEUE FORWARD LINK
```

```
/*KNOWN FILE QUEUE BACK LINK
```

```
/*SIZE OF BLOCK
```

```
/*STRUCTURE TYPE
```

```
/*CONTROL BITS
```

```
/*KNOWN FILE HEADER BLOCK
```

```
/*OPEN BY FILE ID IF SET
```

```
/*DO NOT OPEN THE FILE IF SET
```

```
tag $$; /*SPARE
```

```
/*DELETE AND DO NOT REPLACE ENTRY
```

```
/*ENTRY IS TO BE DELETED
```

```
/*DEVICE UCB OFFSET
```

```
/*NAME THE ABOVE CONSISTENTLY
```

```
/*DIRECTORY NAME STRING OFFSET
```

```
/*FILE NAME STRING OFFSET
```

```
/*FILE TYPE STRING OFFSET
```

```
/*REFERENCE COUNT
```

```
/*KFIQ NUMBER (INDEX INTO VECTOR OF KFIQ'S)
```

```
/*KNOWN FILE ENTRY SEQUENCE NUMBER
```

```
/*LENGTH OF KFI HEADER FIXED PORTION
```

```
/*LENGTH OF KFI HEADER FIXED PORTION
```

```
/*SEQUENCE NUMBER FIELD
```

```
/*FLAGS WORD
```

```
/*KEEP THE IMAGE FILE OPEN
```

```
/*MAKE IMAGE HEADER RESIDENT
```

```
/*MAKE IMAGE SHARED
```

```
/*KNOWN FILE WAS INSTALLED PROTECTED
```

```
tag $$; /*SPARE BITS
```

```
/*LINKABLE IMAGE
```

```
/*USE PROCESS PRIVILEGE MASK
```

```
/*IMAGE HEADER BLOCK IS RESIDENT
```

```
/*IMAGE IS SHARED
```

```
tag $$; /*SPARE BITS
```

```
/*SHARED MEMORY IDENT ALREADY SET
```

```
/*IMAGE IS COMPATABILITY MODE
```



```
GBLSECCNT word unsigned;
USECNT longword unsigned;
WINDOW OVERLAY union fill;
    WINDOW longword unsigned;
    WINDOW_FIELDS structure fill;
        FID_OVERLAY union fill;
            FID word unsigned;
            FID_NUM word unsigned;
        end FID_OVERLAY;
        FID_SEQ word unsigned;
    end WINDOW_FIELDS;
end WINDOW OVERLAY;
IMGHDR OVERLAY union fill;
    IMGHDR longword unsigned;
    FID_RVN word unsigned;
end IMGHDR_OVERLAY;
PROCPRIV quadword unsigned;
MATCHCTL byte unsigned;
FILL 4 byte fill prefix KFIDEF tag $$;
AMECOD word unsigned;
"IDENT" longword unsigned;
constant "LENGTH" equals . prefix KFIS tag K;
constant "LENGTH" equals . prefix KFIS tag C;
```

end KFIDEF;

end_module \$KFIDEF;

```
/*GLOBAL SECTION COUNT IF SHARED
/*USAGE COUNTER

/*WCB ADDRESS IF OPEN

/*FILE ID
/*FILE NUMBER FIELD OF FILE ID
/*FILE SEQUENCE NUMBER FIELD OF FILE ID

/*IMAGE HEADER ADDRESS IF RESIDENT
/*RELATIVE VOLUME NUMBER FIELD OF FILE ID

/*PROCESS PRIVILEGE MASK
/*GLOBAL SECTION MATCH CONTROL
/*SPARE BYTE
/*IMAGE HEADER CODE SPECIFYING AME
/*GLOBAL SECTION IDENT VALUE
/*LENGTH OF FIXED AREA OF KFI ENTRY
/*LENGTH OF FIXED AREA OF KFI ENTRY
```



```
module $KFPDEF;
```

```
/*
```

```
/* KNOWN FILE POINTER BLOCK DEFINITIONS
```

```
/*
```

```
*** obsolete, to be removed ***
```

```
aggregate KFPDEF structure prefix KFP$;
```

```
  QUECOUNT byte unsigned;
```

```
  FILL_1 byte fill prefix KFPDEF tag $$;
```

```
  FILL_2 word fill prefix KFPDEF tag $$;
```

```
  FILL_3 longword fill prefix KFPDEF tag $$;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  TYPE1 byte unsigned;
```

```
  QUEO longword unsigned;
```

```
/*INDEX OF LAST KNOWN FILE LIST IN USE
```

```
/*SPARE BYTE
```

```
/*SPARE WORD
```

```
/*SPARE LONG WORD
```

```
/*SIZE OF POINTER BLOCK IN BYTES
```

```
/*POINTER BLOCK TYPE
```

```
/*TYPE OF STRUCTURE POINTED TO
```

```
/*POINTER TO KNOWN FILE QUEUE 0
```

```
end KFPDEF;
```

```
end_module $KFPDEF;
```



```
module $KFPBDEF;
```

```
/*
```

```
/* KNOWN FILE POINTER BLOCK DEFINITIONS
```

```
/*
```

```
aggregate KFPBDEF structure prefix KFPB$;
```

```
  KFDLST longword unsigned;
```

```
  KFEHSHTAB longword unsigned;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  SPARE byte unsigned;
```

```
  KFDLSTCNT word unsigned;
```

```
  HSHTABLEN word unsigned;
```

```
  constant 'LENGTH' equals . prefix KFPB$ tag K;
```

```
  constant 'LENGTH' equals . prefix KFPB$ tag C;
```

```
/* Device, Directory, Extension (KFD) List Link
```

```
/* Address of Known file name hash table
```

```
/* Size of pointer block in bytes
```

```
/* Pointer block type
```

```
/* spare byte
```

```
/* Number of entries in KFD List
```

```
/* Length of Hash table
```

```
/* Length of pointer block
```

```
/* Length of pointer block
```

```
end KFPBDEF;
```

```
end_module $KFPBDEF;
```



```
module $KFRHDEF;
```

```
/*  
/* KNOWN FILE RESIDENT IMAGE HEADER DEFINITIONS  
/*
```

```
aggregate KFRHDEF structure prefix KFRH$;
```

```
  BUFEND longword unsigned;
```

```
  ALIAS word unsigned;
```

```
  FILL_1 word fill prefix KFRHDEF tag $$;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  HDRVER byte unsigned;
```

```
  constant 'LENGTH' equals . prefix KFRH$ tag K;
```

```
  constant 'LENGTH' equals . prefix KFRH$ tag C;
```

```
  IHD character length 0;
```

```
/* Address of end of known file header
```

```
/* Use secondary name on activation
```

```
/* SPARE BYTE
```

```
/* Size of dynamic structure
```

```
/* Dynamic structure type
```

```
/* Image header version
```

```
/* Length of overhead area
```

```
/* Length of overhead area
```

```
/* Offset to decoded Image Header
```

```
/*  
/* THE REMAINDER OF THIS STRUCTURE CONTAINS THE IMAGE HEADER OF THE  
/* SPECIFIED KNOWN FILE. THE LOCATION KFESL_IMGHDR IN THE KNOWN FILE  
/* ENTRY POINTS KFRH$C_LENGTH INTO THIS STRUCTURE, I.E AT THE IMAGE HEADER  
/* ITSELF.  
/*
```

```
end KFRHDEF;
```

```
end_module $KFRHDEF;
```



```
module $LKBDEF;
```

```
/*+
/* LKB - LOCK BLOCK
/*
/* LOCK BLOCKS ARE USED TO REPRESENT LOCK REQUESTS (ONE BLOCK FOR EACH
/* REQUEST). LOCK BLOCKS HAVE AN ENTRY IN THE LOCK ID TABLE POINTING
/* TO THEM AND ARE LINKED ONTO ONE OF THREE QUEUES IN A RESOURCE BLOCK (RSB)
/*-
```

```
aggregate LKBDEF structure prefix LKB$;
```

```
  ASTQFL longword unsigned; /*AST QUEUE FORWARD LINK
  ASTQBL longword unsigned; /*AST QUEUE BACKWARD LINK
  SIZE word unsigned; /*SIZE OF LKB IN BYTES
  TYPE byte unsigned; /*STRUCTURE TYPE
  RMOD structure byte unsigned; /*ACCESS MODE OF REQUEST
    MODE bitfield length 2; /* MODE SUBFIELD
    FILL 1 bitfield length 2 fill prefix LKBDEF tag $$/ * SPARE
    PKAST bitfield mask; /* PIGGY BACK SPECIAL KERNEL AST
    NODELETE bitfield mask; /* DON'T DELETE ACB ON DELIVERY
    QUOTA bitfield mask; /* ACCOUNT FOR QUOTA
    KAST bitfield mask; /* SPECIAL KERNEL AST
  end RMOD;
  PID longword unsigned; /*PROCESS ID OF REQUESTING PROCESS
  AST_OVERLAY union fill;
    AST longword unsigned; /*ADDRESS OF AST ROUTINE
    RQSEQNM word unsigned; /*REQUEST SEQ. NUMBER
  end AST_OVERLAY;
  ASTPRM_OVERLAY union fill;
    ASTPRM longword unsigned; /*AST PARAMETER
    EPID longword unsigned; /*EPID (MASTER COPIES ONLY)
  end ASTPRM_OVERLAY;
  KAST_OVERLAY union fill;
    KAST longword unsigned; /*SPECIAL KERNEL AST ADDRESS
    DUETIME longword unsigned; /*DUETIME FOR WAITING LOCKS
  end KAST_OVERLAY;
  CPLASTADR longword unsigned; /*ADDRESS OF COMPLETION AST ROUTINE
  BLKASTADR longword unsigned; /*ADDRESS OF BLOCKING AST ROUTINE
  LKSB_OVERLAY union fill;
    LKSB longword unsigned; /*ADDRESS OF LOCK STATUS BLOCK
    DLCKPRI longword unsigned; /*DEADLOCK PRIORITY (MASTER COPIES)
  end LKSB_OVERLAY;
  FLAGS word unsigned; /*USER SPECIFIED FLAGS
  STATUS structure word unsigned; /*INTERNAL STATUS
    DCPLAST bitfield mask; /* DELIVER COMPLETION AST
    DBLKAST bitfield mask; /* DELIVER BLOCKING AST
    ASYNC bitfield mask; /* REQUEST COMPLETED ASYNCHRONOUSLY
    BLKASTQED bitfield mask; /* BLOCKING AST HAS BEEN QUEUED
    MSTCPY bitfield mask; /* LKB IS A MASTER COPY
    NOQUOTA bitfield mask; /* DON'T CHARGE QUOTA
    TIMOUTQ bitfield mask; /* LKB IS ON TIMEOUT QUEUE
    WASSYSOWN bitfield mask; /* WAS SYSTEM OWNED LOCK
    CVTTOSYS bitfield mask; /* CVT BACK TO SYS. OWNED
    PROTECT bitfield mask; /* PROTECTED LOCK
    RESEND bitfield mask; /* RESEND DURING FAILOVER
```



```

end STATUS;
LKST1 longword unsigned;          /*FIRST LOCK STATUS LONGWORD
LKST2 OVERLAY union fill;
    LKST2 longword unsigned;      /*SECOND LOCK STATUS LONGWORD
    constant ACBLEN equals . prefix LKB$ tag K; /*LENGTH OF ACB PORTION OF LKB
    constant ACBLEN equals . prefix LKB$ tag C; /*LENGTH OF ACB PORTION OF LKB
    LKID longword unsigned;       /*LOCK ID
end LKST2 OVERLAY;
RQMODE byte unsigned;             /*REQUEST MODE
GRMODE byte unsigned;             /*GRANTED MODE
STATE byte unsigned;              /*LOCK STATE
constant (
    GRANTED,                      /* GRANTED
    CONVERT,                      /* CONVERSION
    WAITING,                      /* WAITING
    RETRY,                        /* RETRY REQUEST
    SCSWAIT,                     /* SCS WAIT
    RSPNOTQED,                   /* RESPONSE NOT QUEUED
    RSPQUEUED,                   /* RESPONSE QUEUED
    RSPGRANTD,                   /* RESPONSE GRANTED
    RSPDOLOCL,                   /* RESPONSE DO LOCALLY
    RSPRESEND,                   /* RESPONSE RESEND
) equals 1 increment -1;
EFN byte unsigned;                /*EVENT FLAG NUMBER
SQFL longword unsigned;           /*STATE QUEUE FORWARD LINK
SQBL longword unsigned;           /*STATE QUEUE BACKWARD LINK
OWNQFL longword unsigned;         /*OWNER QUEUE FORWARD LINK
OWNQBL longword unsigned;         /*OWNER QUEUE BACKWARD LINK
PARENT longword unsigned;         /*ADDRESS OF PARENT LKB
REFCNT word unsigned;             /*SUB LKB REFERENCE COUNT
TSLT byte unsigned;               /*TIMESTAMP LIFETIME
FILL_2 byte fill;                 /*SPARE
RSB longword unsigned;            /*ADDRESS OF OWNER RSB
REMLKID longword unsigned;        /*REMOTE LOCK ID
CSID OVERLAY union fill;
    CSID longword unsigned;       /*CLUSTER SYSTEM ID (MASTER ONLY)
    OLDASTPRM longword unsigned;  /*OLD AST PARAMETER
end CSID OVERLAY;
OLDBLKAST longword unsigned;      /*OLD BLOCKING AST ADDR,
constant "LENGTH" equals . prefix LKB$ tag K; /*LENGTH OF LKB
constant "LENGTH" equals . prefix LKB$ tag C; /*LENGTH OF LKB
end LKBDEF;

end_module $LKBDEF;
```



```
module $LOGDEF;
```

```
/*+
/* LOG - LOGICAL NAME BLOCK
/*
/* THERE IS ONE LOGICAL NAME BLOCK FOR EACH LOGICAL NAME ASSIGNMENT IN A
/* SYSTEM. LOGICAL NAME BLOCKS CAN BE LINKED INTO ONE OF THREE TABLES:
/*
/* 1. A PER PROCESS TABLE.
/* 2. A GROUP WIDE TABLE.
/* 3. THE SYSTEM WIDE TABLE.
/*-
```

```
aggregate LOGDEF structure prefix LOG$;
```

```
  LTFL longword unsigned; /*LOGICAL TABLE FORWARD LINK
  LTBL longword unsigned; /*LOGICAL TABLE BACKWARD LINK
  SIZE word unsigned; /*SIZE OF LOG IN BYTES
  TYPE byte unsigned; /*STRUCTURE TYPE FOR LOG
  TABLE byte unsigned; /*LOGICAL NAME TABLE TYPE
  GROUP word unsigned; /*CREATOR GROUP NUMBER
  AMOD byte unsigned; /*ACCESS MODE OF CREATOR
  FILL_1 byte fill prefix LOGDEF tag $$; /*SPARE BYTE
  MBXUCB OVERLAY union fill;
    MBXUCB longword unsigned; /*MAILBOX UCB ADDRESS
    constant 'LENGTH' equals . prefix LOG$ tag K; /*LENGTH OF FIXED PART OF LOG
    constant 'LENGTH' equals . prefix LOG$ tag C; /*LENGTH OF FIXED PART OF LOG
  MBXUCB FIELDS structure fill;
    FILL_2 byte dimension 4 fill prefix LOGDEF tag $$;
    NAME character length 0 tag T; /*START OF LOGICAL NAME
```

```
/*
/* LOGICAL NAME TABLE NUMBERS
/*
```

```
  constant SYSTEM equals 0 prefix LOG tag $C; /*SYSTEM NAME TABLE
  constant GROUP equals 1 prefix LOG tag $C; /*GROUP NAME TABLE
  constant PROCESS equals 2 prefix LOG tag $C; /*PROCESS NAME TABLE
```

```
/*
/* MAXIMUM LENGTH OF LOGICAL NAME STRING
/*
```

```
  constant NAMLENGTH equals 64 prefix LOG tag $C; /*MAXIMUM LENGTH OF LOGICAL NAME STRING
```

```
  end MBXUCB FIELDS;
  end MBXUCB_OVERLAY;
end LOGDEF;
```

```
end_module $LOGDEF;
```



```
module $LNMSTRDEF;
```

```
{+
{ LNMB - LOGICAL NAME BLOCK
{
```

```
{ There is one logical name block per logical name that is defined.
{ These block are chained from the hash table ($LNMHSH).
{ Each translation is a sub-block of this structure.
{-
```

```
aggregate LNMDEF structure prefix LNMBS;
```

```
  FLINK longword unsigned; /* Forward link in list
  BLINK longword unsigned; /* Backward link in list
  SIZE word unsigned; /* Size of LNMB in bytes
  TYPE byte unsigned; /* Structure type for LNMB
  ACMODE byte unsigned; /* Owner access mode / integrity level byte
  TABLE longword unsigned; /* Logical name table header address
  FLAG BITS union;
    FLAGS byte unsigned; /* Name attributes
    BITS structure;
      NO ALIAS bitfield mask; /* Do not allow outer mode alias
      CONFINE bitfield mask; /* Do not copy into subprocess
      CRELOG bitfield mask; /* Created with old $CRELOG service
      TABLE bitfield mask; /* This is a table name
      NODELETE bitfield mask; /* Do not allow this table to be deleted
    end BITS;
  end FLAG_BITS;
  NAME character length 1; /* Name string (counted)
                          /* Translation blocks begin immediately
                          /* following name
```

```
end LNMDEF;
```

```
{+
{ LNMC - LOGICAL NAME TABLE NAME CACHE BLOCK
{
```

```
{ There are some logical name table name cache blocks per process.
{ These block are doubly-linked from a P1 space cell (CTL$GQ_LNMTBLCACHE).
{-
```

```
aggregate LNMDEF structure prefix LNMCS;
```

```
  FLINK longword unsigned; /* Forward link in list
  BLINK longword unsigned; /* Backward link in list
  SIZE word unsigned; /* Size of LNMC in bytes
  TYPE byte unsigned; /* Structure type for LNMC
  CACHEINDX byte unsigned; /* Current entry number
  TBLADDR longword unsigned; /* Logical name table name address
  PROCDIRSEQ longword unsigned; /* Process directory sequence number
  SYSDIRSEQ longword unsigned; /* System directory sequence number

  constant NUM_ENTRIES equals 26; /* Number of table header entries.

  ENTRY longword dimension LNMCS$K_NUM_ENTRIES; /* Logical name table header addresses

  constant LENGTH equals . prefix LNMCS$ tag K; /* Length of header
```

```
end LNMDEF;
```



```

{+
{ LNMX - LOGICAL NAME TRANSLATION BLOCK
{
{ There is one logical name translation block per logical name translation.
{ These blocks are sub-blocks of the logical name block ($LNMB).
{-

aggregate LNMXDEF structure prefix LNMX$;
  FLAG BITS union;
    FLAGS byte unsigned;          /* Translation attributes
    BITS structure;               /* Do not display result of translation
      CONCEALED bitfield mask;    /* Do not retranslate result of translation
      TERMINAL bitfield mask;    /* End of translations flag
      XEND bitfield mask;
    end BITS;
  end FLAG_BITS;
  INDEX byte;                    /* Translation index
  constant (
    'HSHFCN',                    /* Hash function value
    'BACKPTR',                  /* Backpointer translation
    'TABLE'                      /* Logical name table header
  ) equals -128 increment 1 prefix LNMX$ tag C;
  HASH word;                    /* Hash code for logical names in directories
  XLATION character length 1;    /* Translation string (counted)
                                  /* The next translation block
                                  /* begins immediately following
                                  /* this translation string

end LNMXDEF;

{+
{ LNMTH - LOGICAL NAME TABLE HEADER BLOCK
{
{ There is one logical name table header block for each logical name table.
{ A logical name table header is a specially flagged translation.
{-

aggregate LNMTHDEF structure prefix LNMTH$;
  FLAG BITS union;
    FLAGS byte unsigned;          /* Logical name table flags
    BITS structure;
      SHAREABLE bitfield mask;    /* Logical name table is shareable (S0 space)
      DIRECTORY bitfield mask;    /* Logical name table is a directory table
      GROUP bitfield mask;        /* Logical name table is a group logical name table
      SYSTEM bitfield mask;       /* Logical name table is the system logical name table
    end BITS;
  end FLAG_BITS;
  HASH longword unsigned;         /* Address of hash table
  ORB longword unsigned;          /* Address of Object Rights Block
  NAME longword unsigned;         /* Address of containing LNMB block
  PARENT longword unsigned;       /* Address of parent table
  CHILD longword unsigned;        /* Address of a child table
  SIBLING longword unsigned;      /* Address of a sibling table
  QTABLE longword unsigned;       /* Address of table holding quota

```



```
        BYTESLM longword;          /* Initial quota
        BYTES longword;           /* Remaining quota
constant LENGTH equals . prefix LNMTH$ tag K; /* Length of header
end LNMTHDEF;

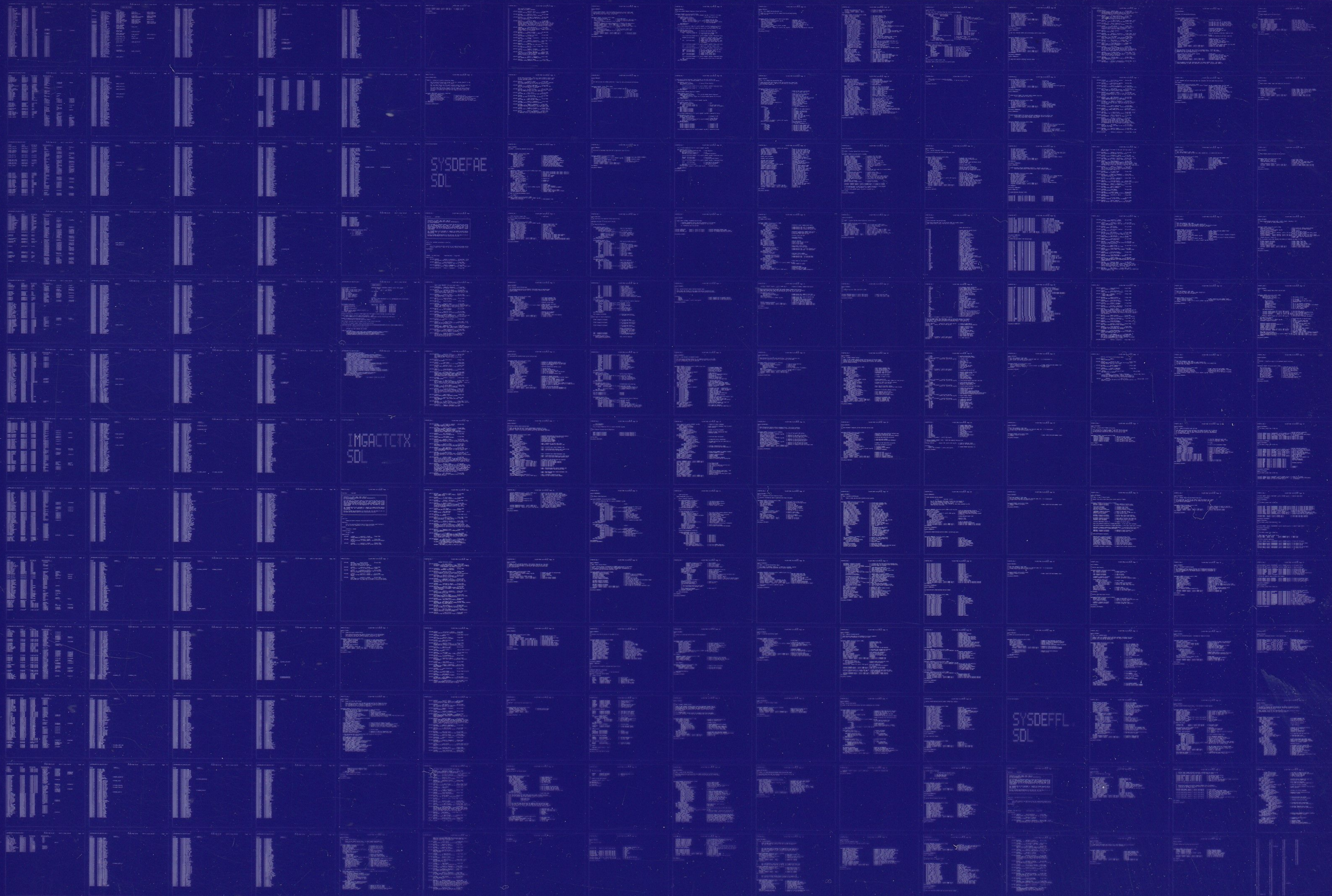
{+
{ LNMHSH - LOGICAL NAME HASH TABLE BLOCK
{
{ There is one logical name hash table block for system space and
{ another for each process.
{-

aggregate LNMHSHDEF structure prefix LNMHSH$;
    MASK longword unsigned;        /* Mask for hash value
    FILL_1 longword fill;          /* Spare longword
    SIZE word unsigned;            /* Size of LNMHSH in bytes
    TYPE byte unsigned;            /* Structure type for LNMHSH
    FILL_2 byte fill;              /* Spare byte
    constant 'BUCKET' equals . prefix LNMHSH$ tag C; /* Length of fixed part of LNMHSH
    constant 'BUCKET' equals . prefix LNMHSH$ tag K; /* Length of fixed part of LNMHSH
end LNMHSHDEF;

end_module $LNMSTRDEF;
```


0370 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0371 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

